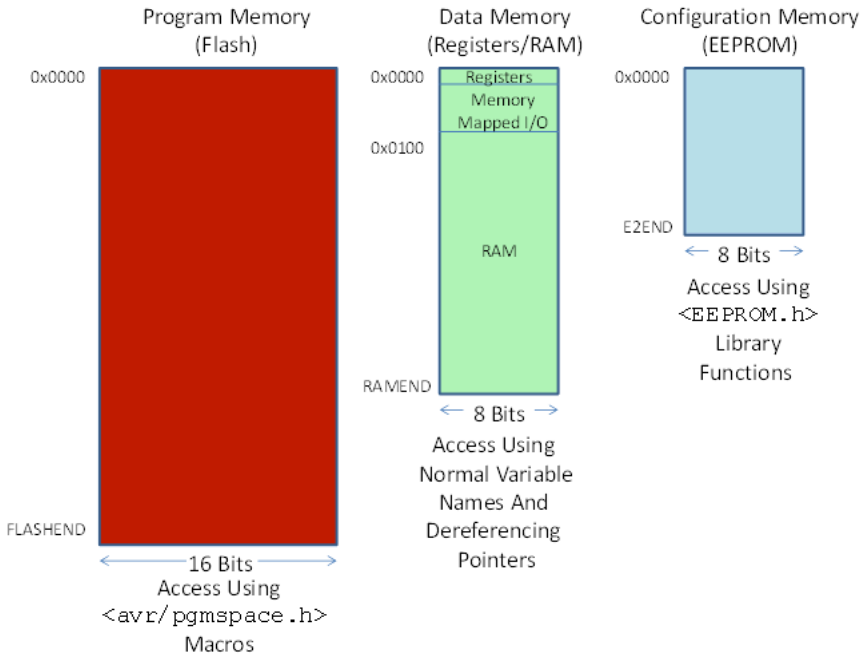
CS 341 – Lab 4
Memory Mapped I/O

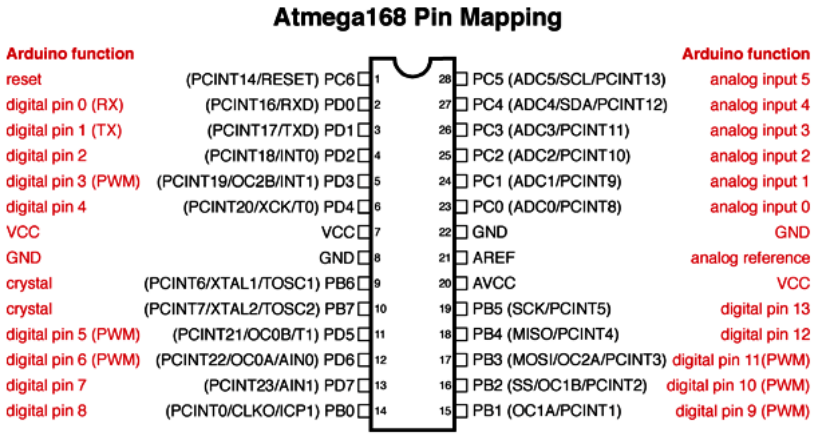
In this lab, you will learn about memory mapped I/O on the Arduino UNO computer. The starter code can be found on GitHub under Lab 4.

Some Background Info

The ATMEGA processor on the Arduino UNO board implements a concept called memory mapped I/O. Instead of using a separate I/O address space and special assembly language instructions such as `in` and `out` to access I/O device registers like the Intel architecture does, the I/O device registers are mapped into memory space. In the ATMEGA Harvard architecture, I/O devices are mapped into the data memory space – not the program memory space. The 64 normal I/O device registers are located in data memory space from location 0x20 to 0x5f. There is also room reserved in the data memory space for 160 extended I/O registers from location 0x60 to 0xff. RAM starts at 0x100.



For each I/O pin on the Arduino (you can find pin out of the ATMEGA368 below), there is a mode bit in a port register that controls whether it is configured as an input or output. If the pin is configured as an input pin, a HIGH level on the pin will be read as a 1 and a LOW level will be read as a 0. If the pin is configured as an output pin, writing a 1 will generate a HIGH level and a 0 for a LOW level.



To program Arduino's I/O pins, we can use library functions provided. For example:

```
int led = 13;          /* define the I/O pin # */
pinMode(led, OUTPUT); /* set the mode for the pin to be output */
digitalWrite(led, HIGH); /* Output a 1 */
```

Since Arduino uses memory mapped I/O, if we know the address of the port and bit definitions that is mapped to a particular pin, we can read or write the state of the pin without using the normal library functions.

If we want to read an input pin's value, we dereference the pointer to read the byte, and use a bit-wise operation to mask the appropriate bit.

If we want to set or reset an output pin's value, we must change the state of only that single bit. We dereference the pointer to read the byte, use bit-wise operations to change the state of the appropriate bit, and dereference the pointer to write the byte back.

Program the I/O pins

The RAM memory addresses used for digital pins 2 through 13 are located between locations 0x20 and 0x2f. Write a sketch with a setup function to display that set of memory locations in HEX. Then, use the library functions to set pin 13 as an output and set it HIGH. Display the memory locations again. Then, use the library functions to set pin 13 LOW and display the memory locations again.

Study those three versions of the state of data memory for the 16 locations designated above. See if you can determine which bits were changed by the library functions. In particular, find the address and data bit for the output state of pin 13. Verify your result with the Standard IO Registers map shown in https://ucexperiment.wordpress.com/2016/03/11/arduino-inline-assembly-tutorial-5-2/. You need to add 0x20 to the Standard IO Registers map shown below to get the right memory address.

Standard IO Registers (0-64):

```
Addr    Name     Bit7   Bit6   Bit5   Bit4   Bit3   Bit2   Bit1
...
0x03   PINB     PINB7  PINB6  PINB5  PINB4  PINB3  PINB2  PINB1  P
0x04   DDRB     DDB7   DDB6   DDB5   DDB4   DDB3   DDB2   DDB1   D
0x05   PORTB    PORTB7 PORTB6 PORTB5 PORTB4 PORTB3 PORTB2 PORTB1 P
0x06   PINC     x      PINC6  PINC5  PINC4  PINC3  PINC2  PINC1  P
0x07   DDRC     x      DDC6   DDC5   DDC4   DDC3   DDC2   DDC1   D
0x08   PORTC    x      PORTC6 PORTC5 PORTC4 PORTC3 PORTC2 PORTC1 P
0x09   PIND     PIND7  PIND6  PIND5  PIND4  PIND3  PIND2  PIND1  P
0x0A   DDRD     DDD7   DDD6   DDD5   DDD4   DDD3   DDD2   DDD1   D
0x0B   PORTD    PORTD7 PORTD6 PORTD5 PORTD4 PORTD3 PORTD2 PORTD1 P
...
```

Be sure to include the address in your report.

Now, add to your sketch some code in the loop function to setup a pointer to the location in memory you determined above. Dereference the pointer to read the byte, flip the state of the appropriate bit with a bit-wise exclusive-or using the mask for the bit that you determined above, dereference the pointer again to write the byte, and delay for about 1 second (1000 milliseconds). If you have done it correctly, the LED on the board should be flashing on and off at the rate you set with the delay call.

If you have time, figure out the address and bit mask for other digital output pins numbered 2 through 12. Demonstrate with an LED and resistor on your simulator (like in Lab 1) that you can use memory mapped I/O to flash an LED attached to one of those pins.

Write your lab report to explain what you did, how you did it, and what you learned about interfacing hardware to a microprocessor and its software (the “sketch”).