**Additional Information for Blinking LEDs Lab**

1. Download Arduino IDE from the following link and install in your computer: <https://www.arduino.cc/en/software>
2. Install Arduino IDE
3. To start, go to File -> New
4. Paste the codes below into the main.c file

#include <avr/io.h> // header file for input output pins

#include <util/delay.h> // header file for delay

#define BLINK\_DELAY\_MS 1000

int main (void)

{

/\* set pin PORTB for output\*/

DDRB =0b11111111;

while (1)

{

/\* set pin high to turn LED on \*/

PORTB=0b00000001;

\_delay\_ms(BLINK\_DELAY\_MS); //delay 1 second

/\* set pin low to turn LED off \*/

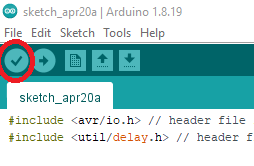
PORTB=0b00000000;

\_delay\_ms(BLINK\_DELAY\_MS); //delay 1 second

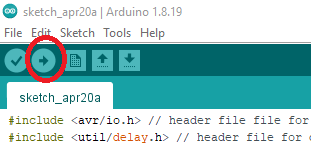
}

}

1. Verify the code by pressing the tick button



1. Connect the Arduino Uno board to the computer through USB.
2. Press the Upload button to upload your program to the Arduino board.

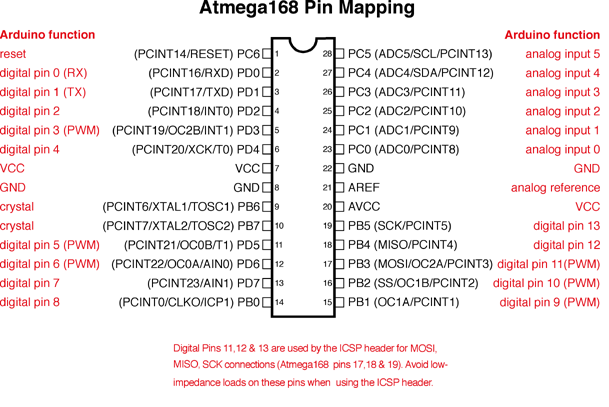


PIN MAPPING:

<https://www.arduino.cc/en/Hacking/PinMapping168>

ATmega168/328P-Arduino Pin Mapping

Note that this chart is for the DIP-package chip. The Arduino Mini is based upon a smaller physical IC package that includes two extra ADC pins, which are not available in the DIP-package Arduino implementations.



PORT REGISTERS:

<https://www.arduino.cc/en/Reference/PortManipulation>

Port registers allow for lower-level and faster manipulation of the i/o pins of the microcontroller on an Arduino board. The chips used on the Arduino board (the ATmega8 and ATmega168) have three ports:

B (digital pin 8 to 13)

C (analog input pins)

D (digital pins 0 to 7)

Each port is controlled by three registers, which are also defined variables in the arduino language. The DDR register, determines whether the pin is an INPUT or OUTPUT. The PORT register controls whether the pin is HIGH or LOW, and the PIN register reads the state of INPUT pins set to input with pinMode(). The maps of the [ATmega8](http://www.arduino.cc/en/Hacking/PinMapping) and [ATmega168](https://www.arduino.cc/en/Reference/Atmega168Hardware) chips show the ports. The newer Atmega328p chip follows the pinout of the Atmega168 exactly.

DDR and PORT registers may be both written to, and read. PIN registers correspond to the state of inputs and may only be read.

**PORTD** maps to Arduino digital pins 0 to 7

DDRD - The Port D Data Direction Register - read/write

PORTD - The Port D Data Register - read/write

PIND - The Port D Input Pins Register - read only

**PORTB** maps to Arduino digital pins 8 to 13 The two high bits (6 & 7) map to the crystal pins and are not usable

DDRB - The Port B Data Direction Register - read/write

PORTB - The Port B Data Register - read/write

PINB - The Port B Input Pins Register - read only

**PORTC** maps to Arduino analog pins 0 to 5. Pins 6 & 7 are only accessible on the Arduino Mini

DDRC - The Port C Data Direction Register - read/write

PORTC - The Port C Data Register - read/write

PINC - The Port C Input Pins Register - read only

Each bit of these registers corresponds to a single pin; e.g. the low bit of DDRB, PORTB, and PINB refers to pin PB0(digital pin 8). For a complete mapping of Arduino pin numbers to ports and bits, see the diagram for your chip: [ATmega8](https://www.arduino.cc/en/Hacking/PinMapping), [ATmega168](https://www.arduino.cc/en/Hacking/PinMapping168). (Note that some bits of a port may be used for things other than i/o; be careful not to change the values of the register bits corresponding to them.)

#### Examples

Referring to the pin map above, the PortD registers control Arduino digital pins 0 to 7.

You should note, however, that pins 0 & 1 are used for serial communications for programming and debugging the Arduino, so changing these pins should usually be avoided unless needed for serial input or output functions. Be aware that this can interfere with program download or debugging.

DDRD is the direction register for Port D (Arduino digital pins 0-7). The bits in this register control whether the pins in PORTD are configured as inputs or outputs so, for example:

DDRD = B11111100; // sets Arduino pins 2 to 7 as outputs, pin 0 and 1 as input

PORTD is the register for the state of the outputs. For example;

PORTD = 0b10101000; // sets digital pins 7,5,3 HIGH

You will only see 5 volts on these pins however if the pins have been set as outputs using the DDRD register or with pinMode().

PIND is the input register variable It will read all of the digital input pins at the same time.