Oct 23, 2021

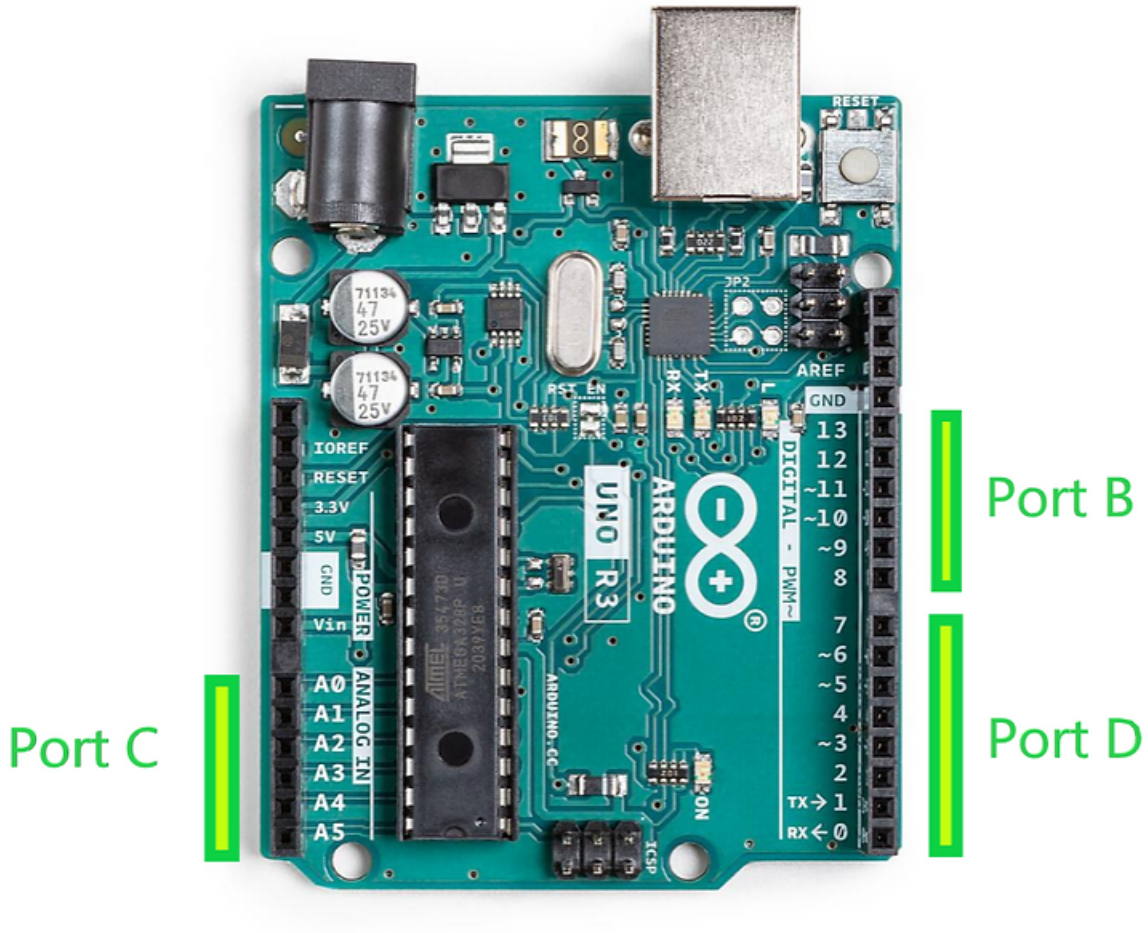
Reading and Writing from and into Arduino Ports

Updated: Oct 25, 2021

In this tutorial, we will be looking at how to Read and Write from and into Arduino Ports.

The reason why I've decided to do this tutorial is that as we go deeper into programming, we will arrive at a point where we need to read or manipulate values of multiple pins whether it is HIGH or LOW seamlessly, often times it would be 8 pins representing 8 bits or a byte of information. For instance, when reading or writing values into LCD display which interfaces with your Arduino module.

Below figure shows all the Ports that we have in Arduino.



- **Port D has Digital Pin 0 to 7**
- **Port B has Digital Pin 8 to 13**
- **Port C has Analog Pin A0 to A5**

The usual method of setting an input or output pin or setting it to high or low value would be to use pinMode and digitalWrite. However, there is another command in Arduino which can perform the same in a much faster way. That would be to use DDR, Port and PIN command.

Taken from the Arduino's website, below would the definition of each commands:

- **The DDR register, determines whether the pin is an INPUT or OUTPUT**
- **The PORT register controls whether the pin is HIGH or LOW**
- **The PIN register reads the state of INPUT pins set to input with pinMode()**

We will be using Port and Pin command in this tutorial to illustrate an example on how we can write and read from an Arduino Port.

For this tutorial, we will be using the code below. This code will construct a 8-bit information or a byte from two Ports which are Port B and D.

```
void setup () {
  Serial.begin(9600);
  uint8_t d = B10101010;
  Serial.print("Inserted Values : ");
  Serial.println(d, BIN);

  /*The PORT register controls whether the pin is HIGH or LOW
  PORT D (digital pins 0 to 7)*/
  PORTD = (PORTD & B00000011) | ((d) & B11111100);

  /*(PORTD & B00000011) = clears PIN 2-7 while maintaining 0 and 1
  ((d) & B11111100) = Write values from "d" */
  //(PORTD & B00000011) | ((d) & B11111100) = final OR maps the value
to PORT D

  /*PORT B (digital pin 8 to 13)*/
  PORTB = (PORTB & B11111100) | ((d) & B00000011);

   /*PIN register reads the state of INPUT pins set to input with
pinMode() */
  uint8_t pin2to7 = PIND & B11111100;
  uint8_t pin0to1 = PINB & B00000011;
  uint8_t pin0to7 = pin2to7 | pin0to1;
  Serial.println("BEGIN");
  Serial.print("pin2to7 :");
  Serial.println(pin2to7, BIN);

  Serial.print("pin0to1 :");
  Serial.println(pin0to1, BIN);
```

```
  Serial.print("pin0to7 :");
  Serial.println(pin0to7, BIN);
}

void loop () {
}
```

Now, let's study this program.

```
  uint8_t d = B10101010;
```

This part of the code basically tells that we're setting a byte called 'd' and its value would be 10101010 which is in binary format.

```
  /*The PORT register controls whether the pin is HIGH or LOW
  PORT D (digital pins 0 to 7)*/
  PORTD = (PORTD & B00000011) | ((d) & B11111100);
```

What happens next is, value 'd' is written into Port D.

(PORTD & B00000011) clears Pin 2 to 7 by setting it to 0 while allowing Pin 0 and 1 to maintain its original state. **The AND logic used here allow Pin 0 and 1 to maintain its original value**.

For an example, let's say current value Port D = **11010101**

Port D = 11010101 then **PORTD & B00000011 = 00000001**

We wouldn't want to change the values of Pin 0 and 1 as it will be used by Arduino when uploading program. By doing the above AND operation, we can see that the original value of Pin 0 and 1 remains the same regardless of what values we have in Pin 2 to 7.

(d) & B11111100) writes the value of 'd' binaries except for the first two bits which will always be 0. For instance, let's say d = **11001101**

If **d = 11001101** then **(d) & B11111100) = 11001100**

Then, PORTD = (PORTD & B00000011) | ((d) & B11111100) operation will write bit 2 to 7 in byte 'd' to PIN 2 to 7 in Port D while retaining the original state of Pin 0 and 1.

00000001 | 11001100 then Port D = **11001101**

Moving on, the same operation happens in Port B.

```
/*PORT B (digital pin 8 to 13)*/
PORTB = (PORTB & B11111100) | ((d) & B00000011);
```

The only difference this time is that byte 'd' is ONLY written to Pin 8 and 9.

For instance, let's say Port B = **00010100** and **d = 11001101**

(PORTB & B11111100) = 00010100
(d) & B00000011 = 00000001
00010100 | 00000001 then Port B = **00010101**

At this stage, we already have **Bit 0 and 1 from byte 'd' embedded into Port B** and **Bit 2 to 7 embedded into Port D**.

Now, let's move onto the next part of our program.

```
uint8_t pin2to7 = PIND & B11111100;
uint8_t pin0to1 = PINB & B00000011;
```

The above line simply read bit 0 and 1 from Port B and bit 2 to 7 from Port D. Earlier we have established that Port B = **00010101** and Port D = **11001101.** Hence,

uint8_t pin2to7 = **11001101** & B11111100 = 11001100
uint8_t pin0to1 = **11001101** & B00000011 = 00000001

Finally, to get back the initial value 'd' that we wrote into Port B and D, we use the below operation

```
uint8_t pin0to7 = pin2to7 | pin0to1;
```

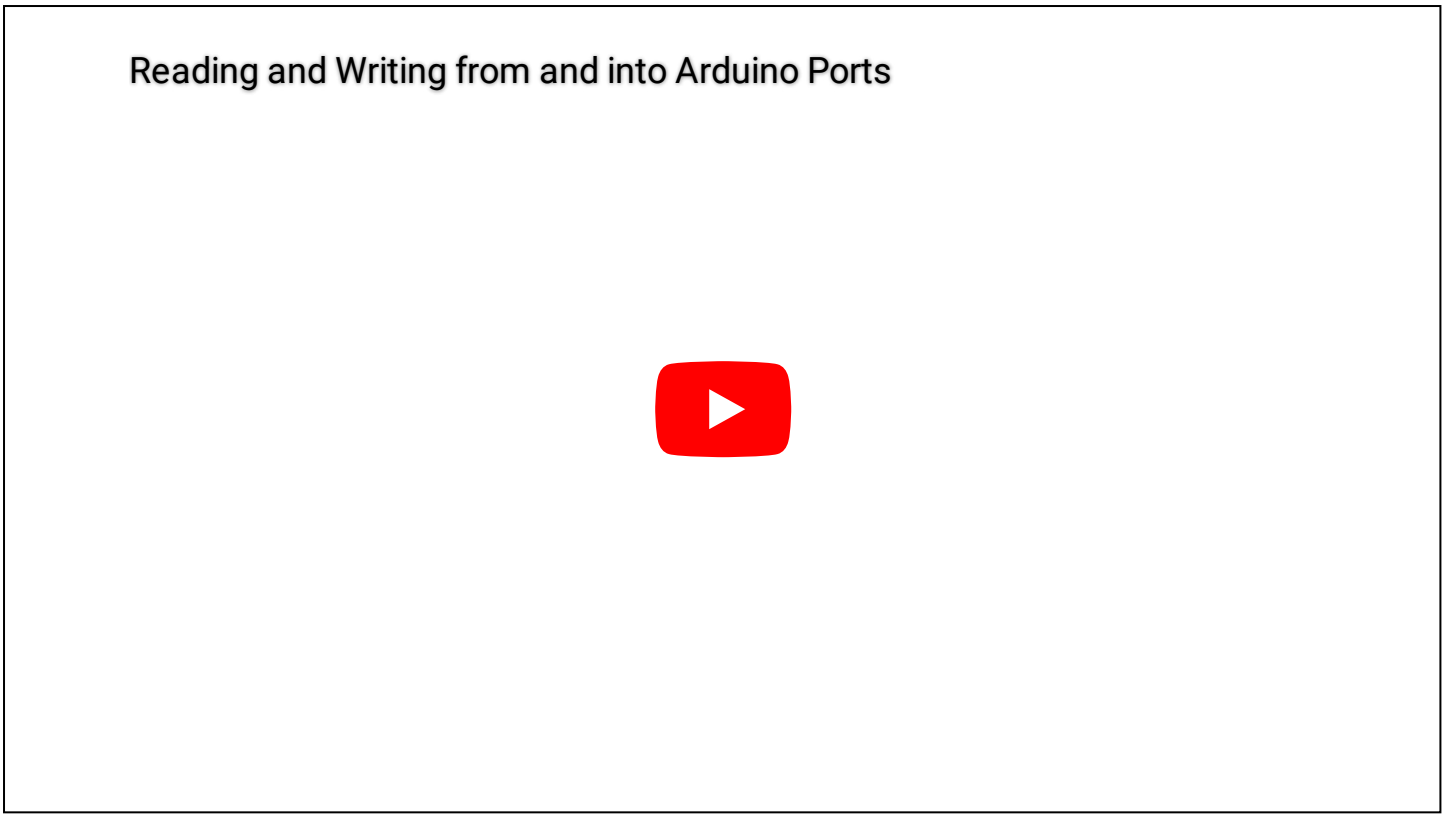
uint8_t pin0to1 = 11001100 | 00000001 = **11001101**

As you can see, the above value is similar to the 'd' value that we wrote into Port B and D. The above program will work for whatever 'd' value as long as it has 8 bits.

With this, we have come to the end of tutorial. I hope thru this tutorial you can learn how you can read, write and manipulate Ports to construct 8 bits or a byte of information which will be useful in many application. This tutorial will serve as a fundamental illustration of 8 bit information reading and writing.

Thank You. Please Don't Forge to Like and Subscribe.

Check Out my YouTube video below :


Reading and Writing from and into Arduino Ports