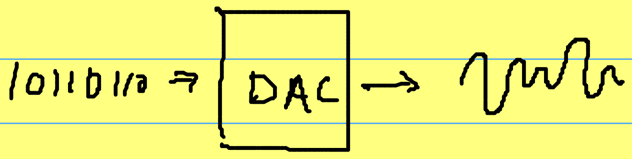
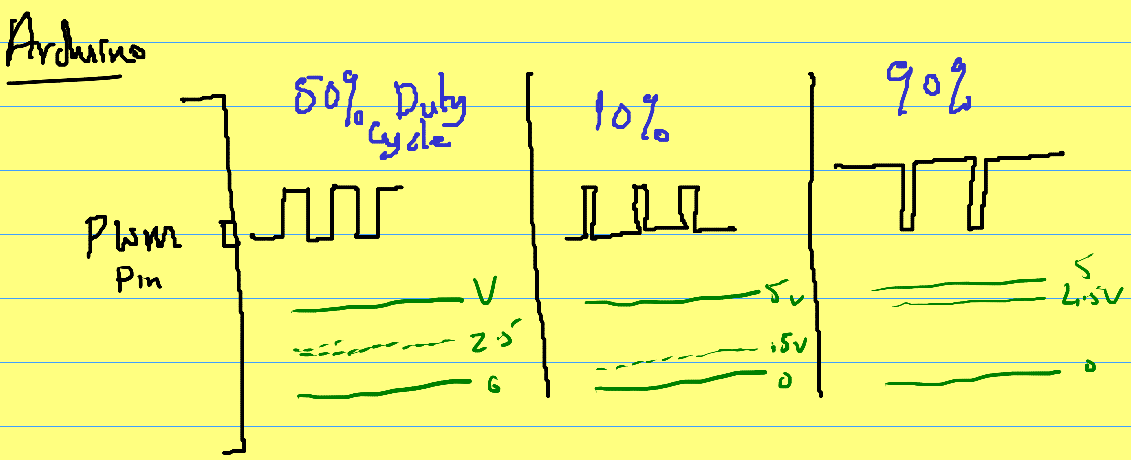
Lecture 6 CS3514 28/09/17

The Arduino has a ADC onboard but it does not have a DAC (Digital to Analog Converter)



Instead of producing a true analog Signal from a DAC, the Arduino has the ability to produce a Pulse-width Modulated (PWM) Signal from Certain pins: (Pins 3,5,6,9,10,11)

A PWM Signal, applied to Certain devices, Simulates the effect of an analog Signal. This effect is achieved when the device being Controlled has an inherent characteristic which allows it to average the energy being applied to it.



By changing the Duty Cycle we can vary the average output from $0_V \rightarrow 5_V$

The microprocessor pin outputting a PWM signal will continually vary from high → low → high. However, the relative length of time the pin stays low compared to the time it spends high is a measure of the average energy emitted by the pin.

The length of a PWM cycle can be thought of as 255 units long. If we write out the value 255 through the PWM pin (using the analogWrite() function), that pin will be high for the whole cycle. That is, it will be high all the time.

In general, if we write out the value x through the PWM pin where $0 \leq x \leq 255$, then the pin will be high for a fraction of the cycle $= \frac{x}{255}$

So for $x = 128$, the pin will be high for half of the cycle time.

If the device to which the PWM pin is connected can 'smoothen out' these impulses in some way, then varying the PWM signal will appear to have an "analog effect" on the device.

For example:

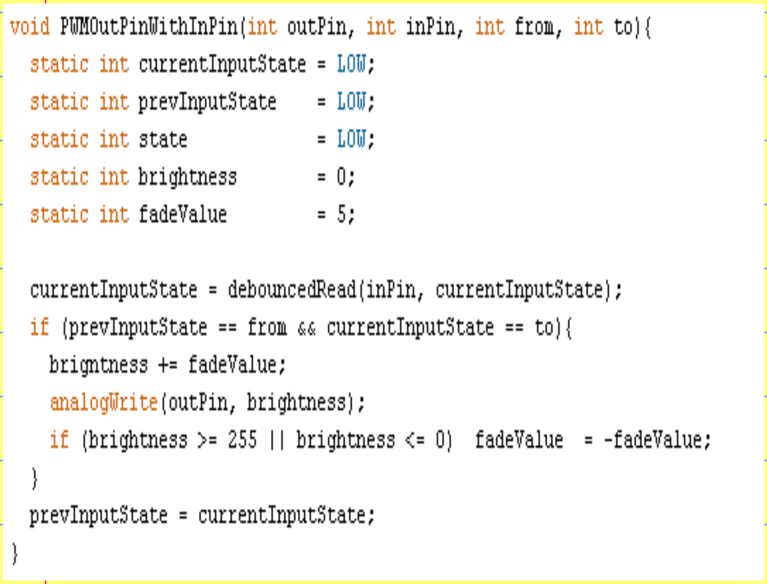
(1) If the output device is an electric motor, the inertia of the rotation of the motor will have the desired smoothing effect and the Duty cycle of the PWM pin (ie. the $\frac{x}{255}$ ratio) will determine the speed of the motor in an analog fashion.

(2) Applying a PWM signal to a LED will cause it to flash rapidly. There is no smoothing in the LED itself to average out the PWM signal — the LED is either on or off. However, our eyes do have a smoothing effect since they cannot transition as quickly as the LED. This smoothing effect - known as Persistance of Vision results in us seeing the LED at various brightnesses (corresponding to the duty cycle) even though the LED itself is always either fully on or fully off!

analogWrite ( ) function

Unfortunately, the name given in the Arduino to the function which generates a PWM Signal on a Pin is

analogWrite( ) — this is somewhat of a misnomer



```
void PWMOutPinWithInPin(int outPin, int inPin, int from, int to){
  static int currentInputState = LOW;
  static int prevInputState    = LOW;
  static int state             = LOW;
  static int brightness        = 0;
  static int fadeValue         = 5;

  currentInputState = debouncedRead(inPin, currentInputState);
  if (prevInputState == from && currentInputState == to){
    brigntness += fadeValue;
    analogWrite(outPin, brightness);
    if (brightness >= 255 || brightness <= 0)  fadeValue  = -fadeValue;
  }
  prevInputState = currentInputState;
}
```

PWM example code to Progressively brighten and dim an LED on successive button presses

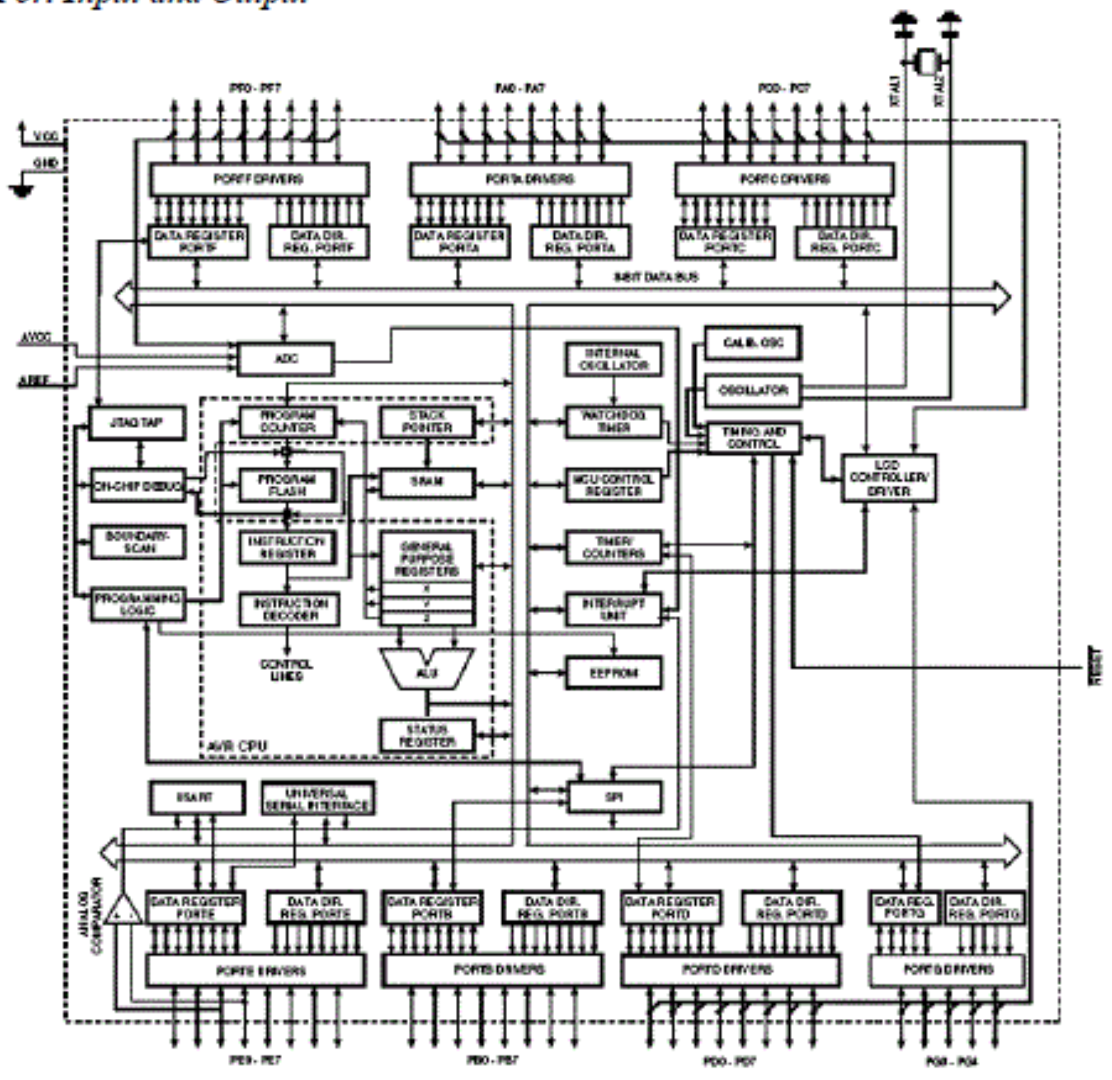
The analogWrite( ) takes 2 Parameters :

(1) The PWM pin

(2) a value from 0 — 255 representing the Duty cycle.

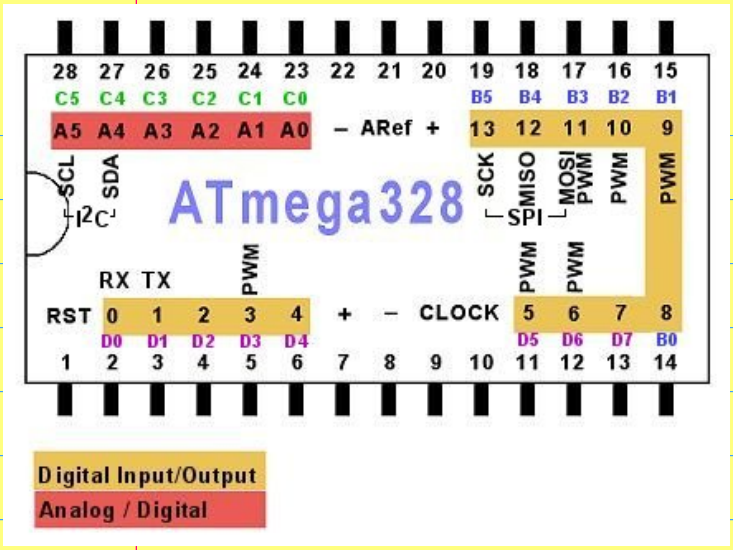
Port Manipulation

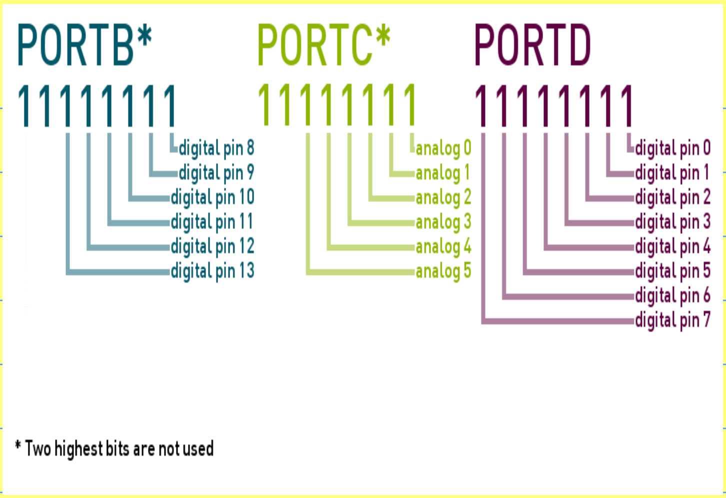
Port Input and Output



Why?

- Speed
- Simultaneous read/write to multiple pins in the same port
- Reduce size of program





* Two highest bits are not used

3 ports and 3 Port Registers — Values in these registers alter the status of the digital & analog I/O pins.

3 ports: D – Digital pins 7 → 0 (8 pins)

remember pins 0,1 are used for Rx Tx.

B – Digital pins 13 → 8 (6 pins)

C – Analog Pins A5 – A0 (6 pins)

To setup a Port for input/output

Up to now we used PinMode(pin, INPUT|OUTPUT);

A special register is used to hold the pin 'direction' for each pin in a port: DDRx Register, where x = D|B|C i.e. We have DDRD, DDRB, DDRC registers

To set the pin directions, we can write

$$\mathrm{DDRD} = B\underset{\uparrow\ \mathrm{MSB}}{x}xxxxxx\underset{\uparrow\ \mathrm{LSB}}{x} \quad \text{where } x = 0|1$$

X = 0 ⇒ Corresponding pin is Input pin

X = 1 ⇒ " " " output "

Example :

$$\mathrm{DDRD} = B\overbrace{1011}^{B}\overbrace{0001}^{1} \equiv \mathrm{DDRD} = 0x\,B1$$

will set pin 7 to output
6 " Input
5 " output
⋮
0 " output

<u>note!</u> The C-language does not have a type 'Binary'
Bxxxxxxxx is, in fact, a predefined constant in the Arduino library.

We could use Hex instead

for example DDRD = 0x0F

This would set pins 7,6,5,4 as Inputs and
pins 3,2,1,0 as outputs

In the C language 0x Indicates the start of a hex constant