ECSESS RoboElectronics

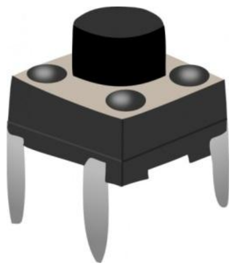
Understanding Inputs and Outputs (I/O)

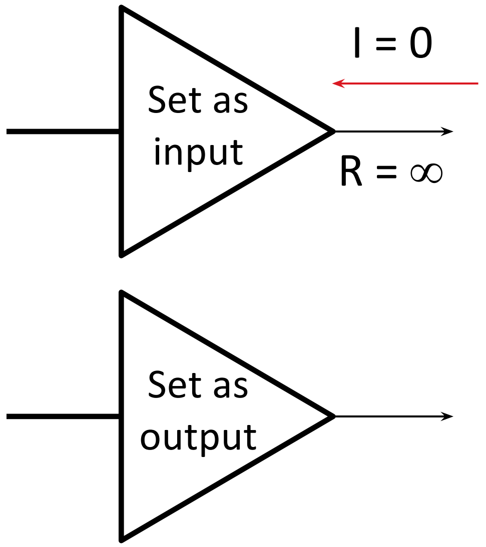
Topics Covered

- Setting ports as an input or output
- Pull up resistors and reading buttons
- Using a 7 segment displays

Inputs and Outputs

- Micro controllers are very flexible and let us set which pins we want to be an input or an output.
- A tristate register allows us to set whether a pin is an input or an output.
- When a pin is set as an output, we can drive a voltage and current on that pin
- When a pin is set as input, it is in an high impedance state, and no current flows into the pin





Configuring Tristate registers

- A tristate register on the 16F88 controls the input or output state of each pin
- Much like the PORTA and PORTB defines, we can use the TRISA and TRISB registers to setup our pins
- The 16F88 datasheet indicates what value sets the pin as an input or output

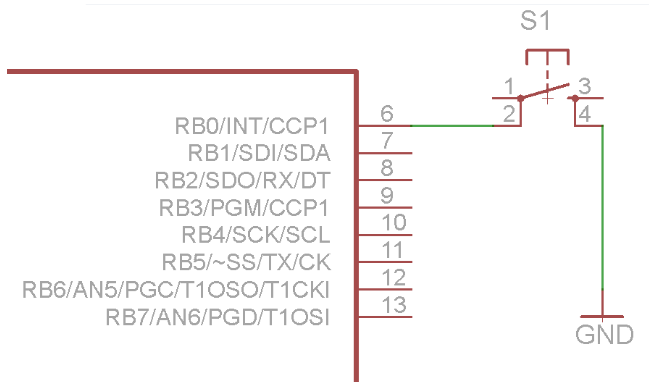
```
#define INPUT 1
#define OUTPUT 0

void init(void)
{
   TRISAbits.TRISA3 = INPUT;
   TRISBbits.TRISB1 = OUTPUT;
}
```

The above example sets pin RA3 as an input and RB1 as output

Buttons and Pull Up resistors

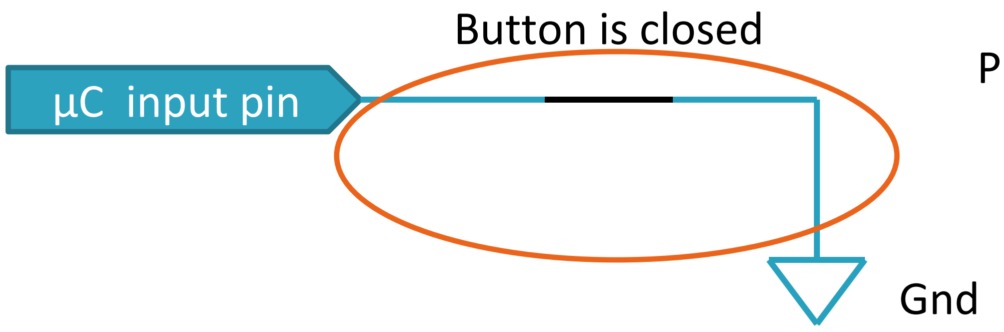
- Buttons are a simple way to provide input to your system
- You can use buttons to do almost anything
  - Select a mode
  - Change a counter
  - Anything you can code
- However, wiring a button to a controller has a few implications
  - Simply connecting a button directly between a pin and ground will not work



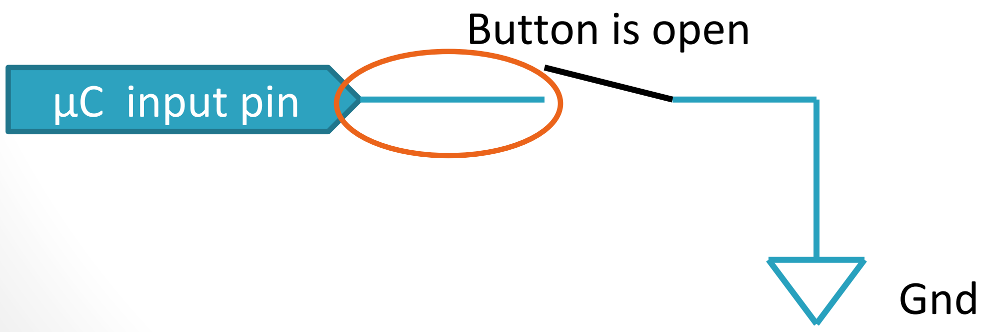
The following image shows a naïve hookup of a button to a µC

Buttons and Pull Up resistors

- Why does the hook up in the previous slide pose problems?
- Recall our µC can only read a value of 0 or 1, LOW or HIGH
- Do a simple circuit analysis



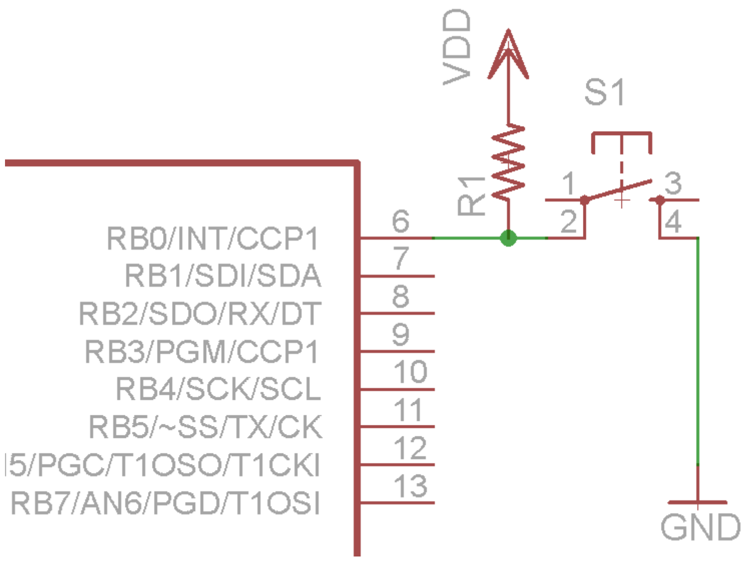
Pin is connected to ground so the voltage is 0



Pin is floating, wires act like antennas, could be reading **any** voltage.
How do we tell if the button has been pushed?

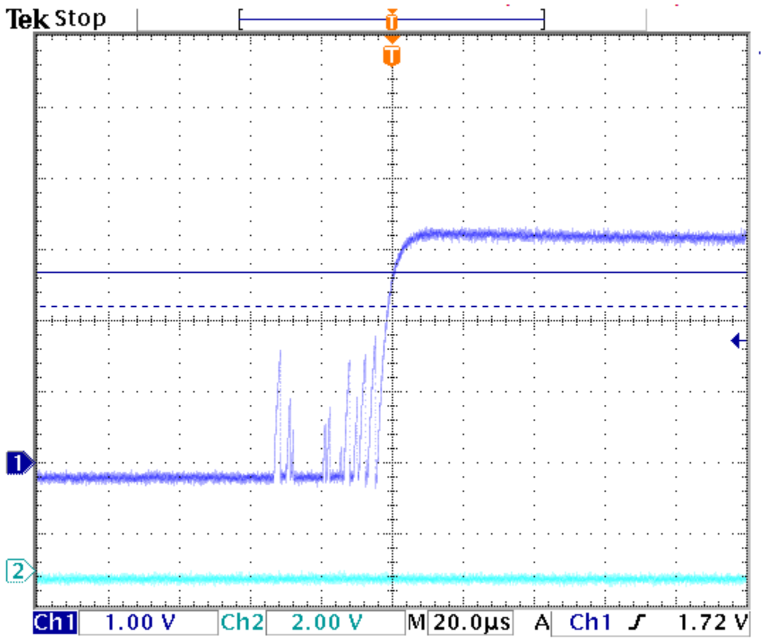
Buttons and Pull Up resistors

- The solution is to add a pull up resistor
- This configuration assures that the pin is never floating
- When the button is open the voltage seen by the pin is VDD
  - A pin is considered to be high impendence, i.e. no current flows into it.
  - This means that there is no current flowing, so there is no voltage drop
- When the button is closed, the voltage seen by the pin in GND
  - Pressing the button connects the resistor to ground
  - This means a current will flow
  - Choosing a high value for R, say 10kΩ assures that only a small current flows



Debouncing buttons

- Since a button is a physical device, its transition from open to closed won't be instantaneous
  - This means there will be some jitter or noise when the button is operated.
- Since we only want to register one button push, we can use a delay to avoid accidentally latching multiple pushes.



The voltage measured at the output of a button where closing the button brings the voltage high

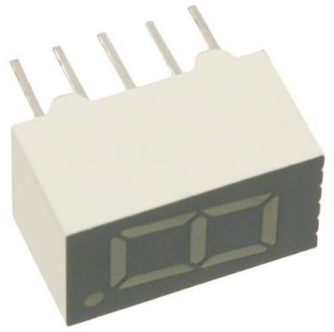
Code to read a button

- Reading a button can be done by configuring your pin to be an input
  - See earlier slides for details
- After simply read the value on the pin
- An if statement is one way to read a button and act on it
  - Note when a button is pushed in our setup it goes to GND
  - Which statement corresponds to the button being pushed?
  - Where would a delay go in this setup?

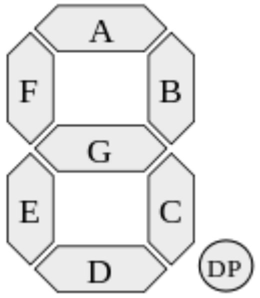
```
#define BUTTON PORTBbits.RB0
.
.
.
if(BUTTON == 1)
{
    //Do one thing
}

if(BUTTON == 0)
{
    //Do another thing
}
```

7-Segment displays



- A 7 segment display uses a set of LEDs to display a number from 0 to 9
- They are nothing more then a set of LEDs in a premade package
- In order to map each LED to a pin, each "segment" is assigned a letter
- All 7 segment packages use the same lettering of segments



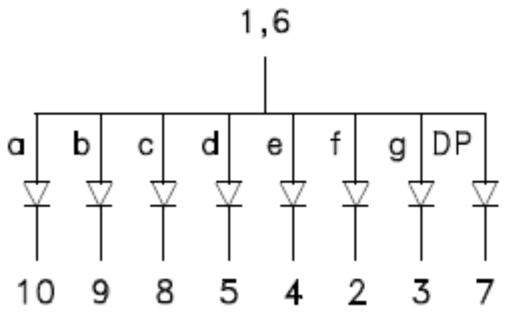
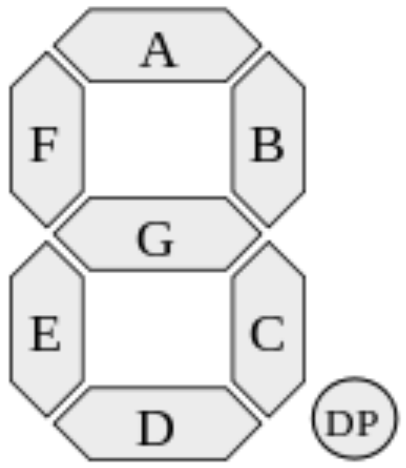
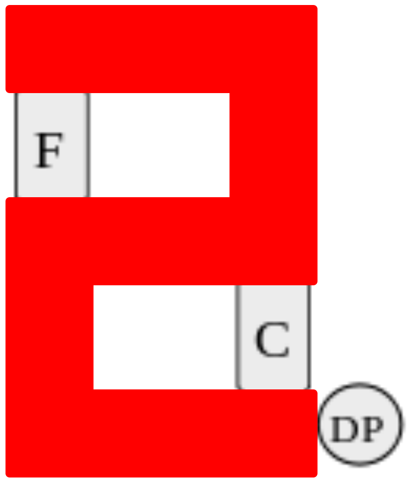


Diagram showing the connection of each led to its corresponding pin

Displaying a number with a 7-Segment

- By turning on certain LEDs, we can select what number we want to display
- For example, turning on A,B, G,E, & D, we will display the number 2
- By connecting the PIC16F88 to the 7-segment, we can display any number from 0 to 9 by turning the LEDs on and off





Code to display a number

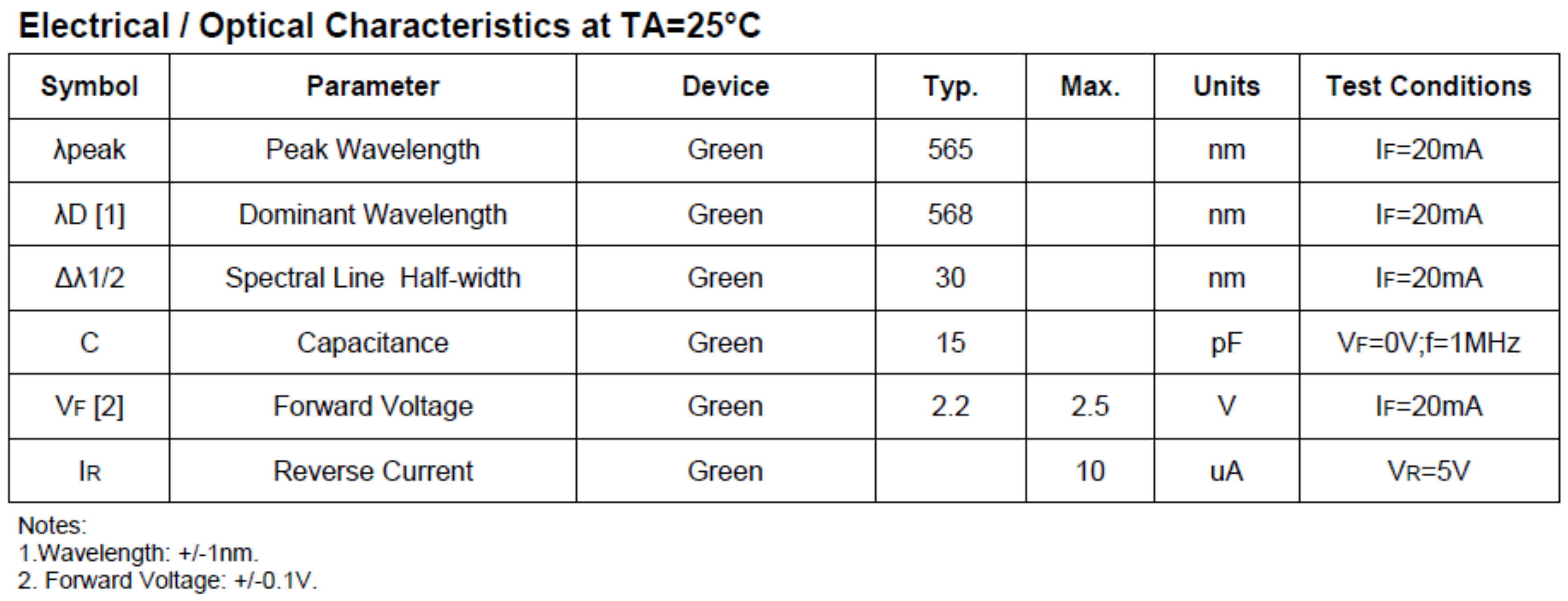
- A define is used to indicate the connection between the pin and segment
- A function then takes in a number and checks which segments to turn on and off
- Recall from the datasheet that the LEDs are common VDD
  - To turn them on you will need a path to ground
- Do you see what is missing from the code?

```
#define LED_A PORTBbits.RB0
#define LED_B PORTBbits.RB1
#define LED_G PORTBbits.RB7
.
.
.

void displayNumber(unsigned int
number)
{
    if(number == 0)
    {
         LED_A = 0;
         LED_B = 0;
         LED_G = 1;
    }
    if(number == 1)
    {
         .
         .
    }
}
```

7-Segments continued

- Remember, just like a regular LED, the ones inside the 7 segment have fixed voltage and current restrictions
- Resistors must be added to their connections accordingly
- Below is the electrical characteristics table from the datasheet



Electrical / Optical Characteristics at TA=25°C

Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λpeak	Peak Wavelength	Green	565		nm	I_F=20mA
λD [1]	Dominant Wavelength	Green	568		nm	I_F=20mA
Δλ1/2	Spectral Line Half-width	Green	30		nm	I_F=20mA
C	Capacitance	Green	15		pF	V_F=0V;f=1MHz
V_F [2]	Forward Voltage	Green	2.2	2.5	V	I_F=20mA
I_R	Reverse Current	Green		10	uA	V_R=5V

Notes:
1.Wavelength: +/-1nm.
2. Forward Voltage: +/-0.1V.

Resources

- Datasheets
  - PIC16F88: http://ww1.microchip.com/downloads/en/DeviceDoc/30487D.pdf
  - PIC16F88 XC8 header file (on your local machine): C:\Program Files (x86)\Microchip\xc8\v1.12\include\pic16f88.h
  - 7-segment display: http://www.kingbrightusa.com/images/catalog/SPEC/SA36-11GWA.pdf
- Further Reading
  - Sparkfun Tutorials: https://learn.sparkfun.com/tutorials/pull-up-resistors