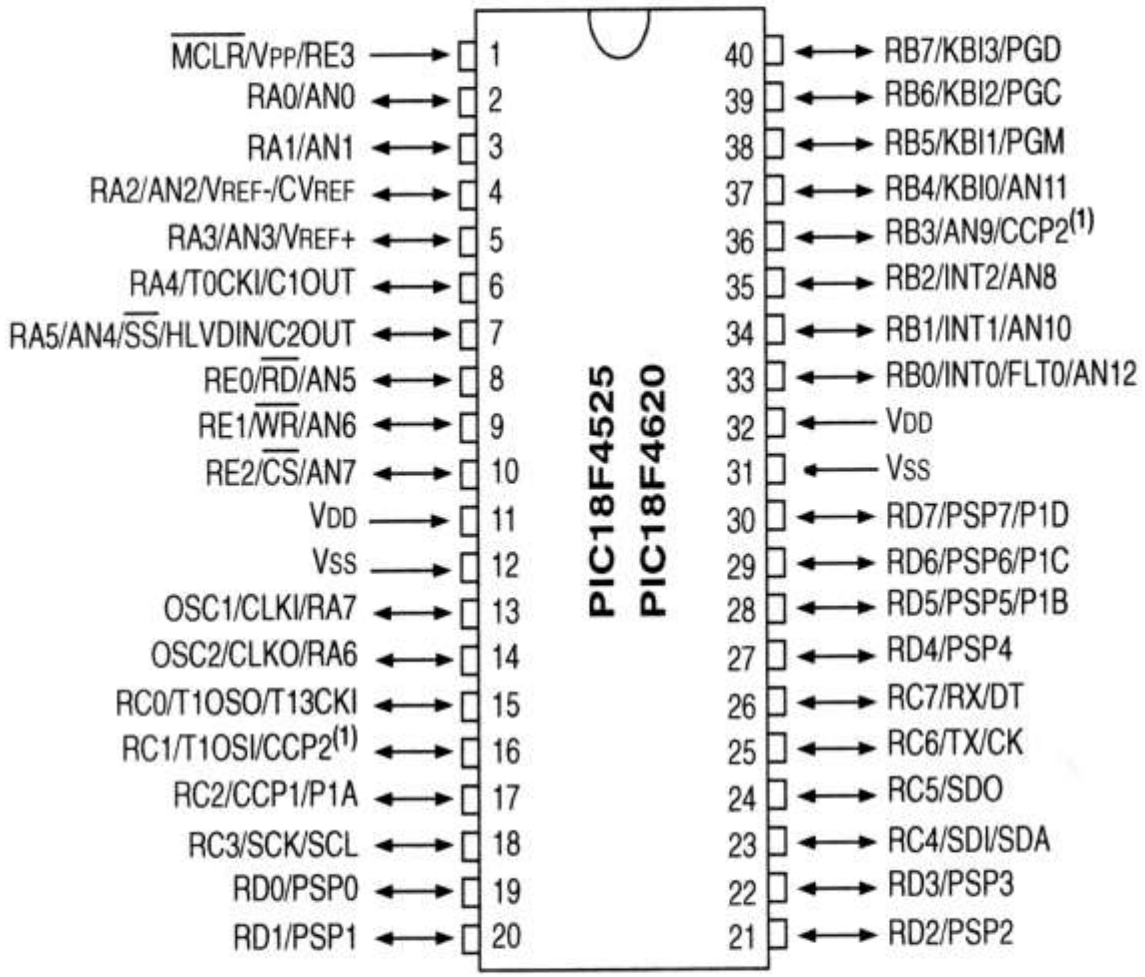
DIO

- A pin can output +5 V (high) or 0 V (low)
- A pin can act as a simple voltmeter and read the voltage at the pin as high (+5 V) or low (0 V)
- PIC18F4525 has 4 ports (groups of 8ish DIO enabled pins): PORTA, PORTB, PORTC, and PORTD.



RAn, RBn, RCn, RDn n = 0 to 7

DIO SFRs

Each port controlled by 8-bit wide Special Function Registers

A	TRISA	PORTA
B	TRISB	PORTB
C	TRISC	PORTC
D	TRISD	PORTD

TRIS

- Sets whether the pins are input (1 bit voltmeter) or an on/off output

$$\texttt{TRISA = 0b}\underbrace{\overset{\text{RA7}}{\overset{\downarrow}{\texttt{0}}}\texttt{00}}_{\text{outputs}}\underbrace{\texttt{11}}_{\text{inputs}}\underbrace{\texttt{00}\overset{\text{RA0}}{\overset{\downarrow}{\texttt{0}}}}_{\text{outputs}}\texttt{;}$$

PORT – Setting Voltage

- Sets output pins high (1 = +5V) or low (0 = 0 V).
- Holds the current voltage at pin if pin is an input.

$$\texttt{PORTA = 0b1}\overset{\text{high}}{\downarrow\atop\texttt{1}}\texttt{1}\underbrace{\texttt{01}}_{\uparrow}\texttt{0}\overset{\text{low}}{\downarrow\atop\texttt{0}}\texttt{0;}$$

Does nothing since inputs. Holds current voltages at input pins.

PORT - Reading Voltage Now

```
Unsigned char RA3_V, RA4_V;

// Check voltage at pins RA3 and RA4
RA3_V = (PORTA & 0b0000 1000) >> 3 ;
// = 1 if +5 V at RA3 else = 0
RA4_V = (PORTA & 0b0001 0000) >> 4 ;
// = 1 if +5 V at RA3 else = 0
```

A Better Way

- Clumsy and cumbersome to work with full port, especially if you want to set or read one or two pins
- Bitfield also provided
- Structures TRISXbits and PORTXbits where X is A, B, C, or D
- TRISXbits.RXn and PORTXbits.RXn address bit level

```
#include <p18f4525.h>   //SFR info for this chip

void main(void)
{
 TRISAbits.RA3 = 1;          // set RA3 as intput
 TRISAbits.RA2 = 0;          // set RA2 as output
 TRISAbits.RA1 = 0;          // set RA1 as output
 PORTAbits.RA2 = 1;          // set RA2 high
 PORTAbits.RA1 = 0;          // set RA1 low
 while(1) {
      // what is RA3 reading now?
      printf(“RA3 voltage (0 = 0V, 1 = +5V) =  %u”,   PORTAbits.RA3);
  }
}
```

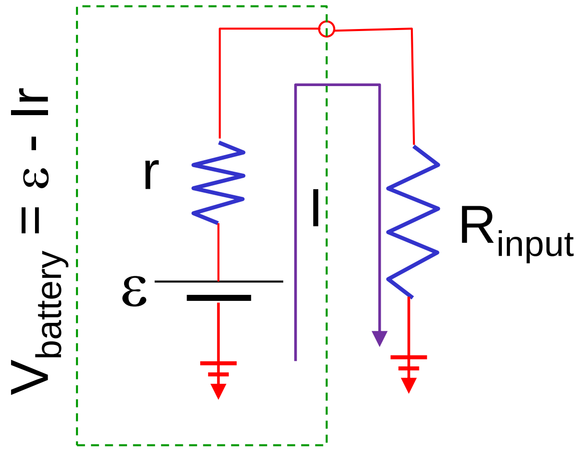
```
main( void )
{
    TRISA = 0b01010011;
    TRISCbits.RC2 = 0;
    TRISCbits.RC4 = 0;
    TRISBbits.RB0 = 1;
    TRISBbits.RB3 = 0;
    TRISD = 0b00111011;
    PORTAbits.RA0 = 0;
    PORTAbits.RA2 = 0;
    PORTBbits.RB3 = 1;
    PORTC = 0b00010000;
    PORTDbits.RD0 = 1;
    PORTBbits.RB0 = 0;
    PORTDbits.RD2 = 1;
    PORTDbits.RD3 = 1;
    while(1);
}
```

Pin	Port Bit	TRIS Bit Value	Port Bit Value	Pin Potential
2	RA0	1	1	5 V
	RA1			0 V
	RA2			
33				0 V
36				
	RC2			
	RC4			
19				0 V
20				5 V
21				
22			0	

Pin Resistance (Impedance)

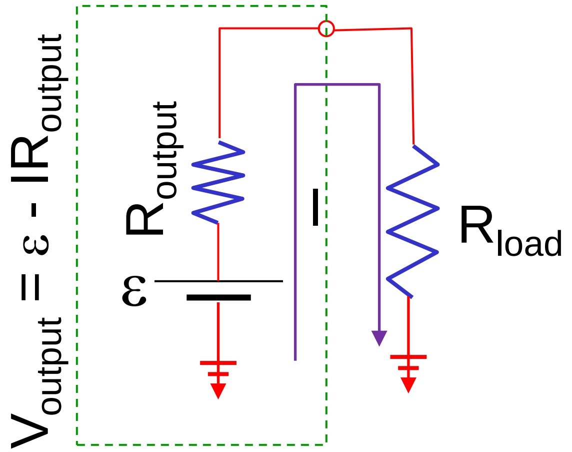
- An input pin has/needs a largish resistance ~ 5 KΩ to act as a voltmeter

$$V_{input} = Ir = \frac{\varepsilon}{R_{input} + r}$$



- An output pin has/needs a small resistance ~ 50 Ω to act as a “battery”

$$V_{load} = Ir = \frac{\varepsilon}{R_{output} + R_{load}}$$



Warning

An output pin at +5V can only produce a maximum current of ~25 mA. Drawing too much current can cause strange behaviour and may damage the chip.

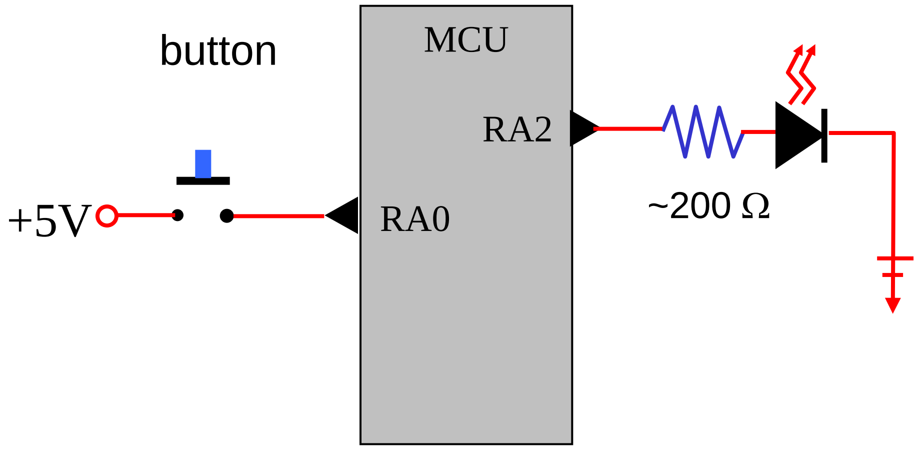
Ohm’s law: $5V/25\ mA = 200\ \Omega$

$\therefore$Connect at least $R_{load} \sim 200\ \Omega$ to any output pin

Input - Buttons

- As a voltmeter, input pin sucks
- As a sensor it can be quite useful if I can program PIC to do something when a pin is high or low.
- Need a circuit that will set input pin to +5 V when a button is pressed and 0 V otherwise

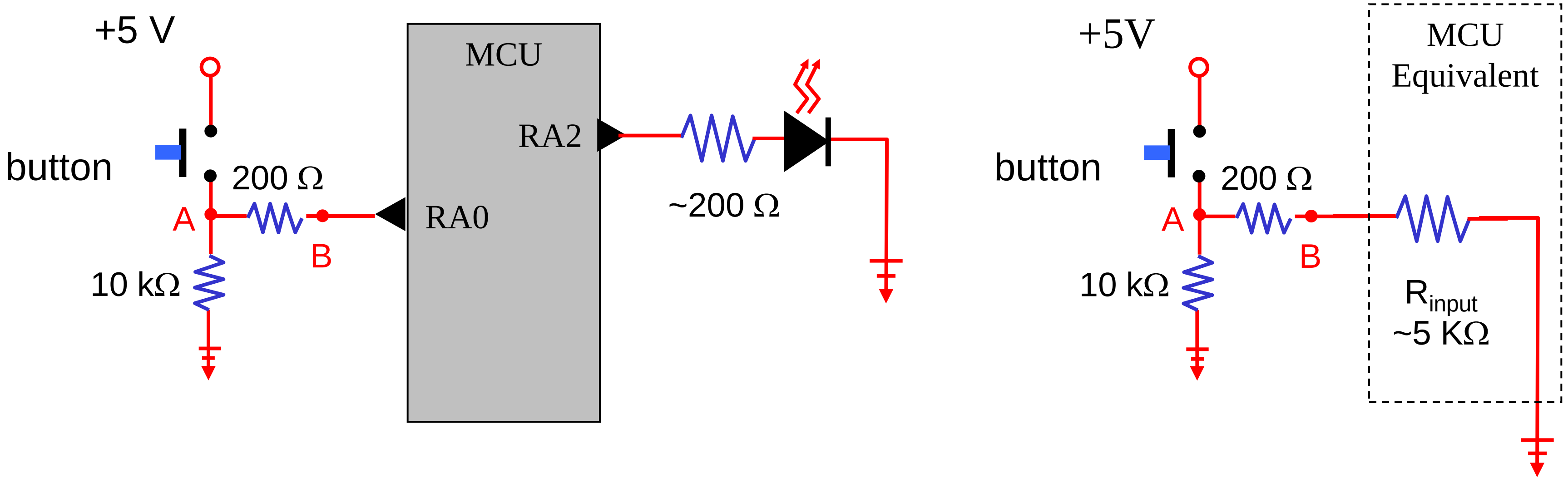
The Wrong Way!!!



Cannot send large currents into input pin. Can damage MCU.

Use a “logic” switch instead.

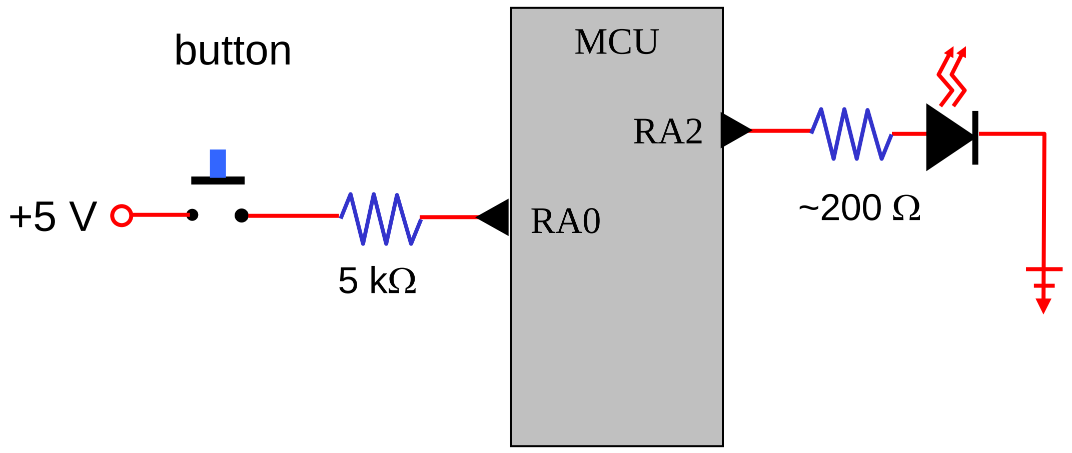
The Right Way



Input has a large resistance.

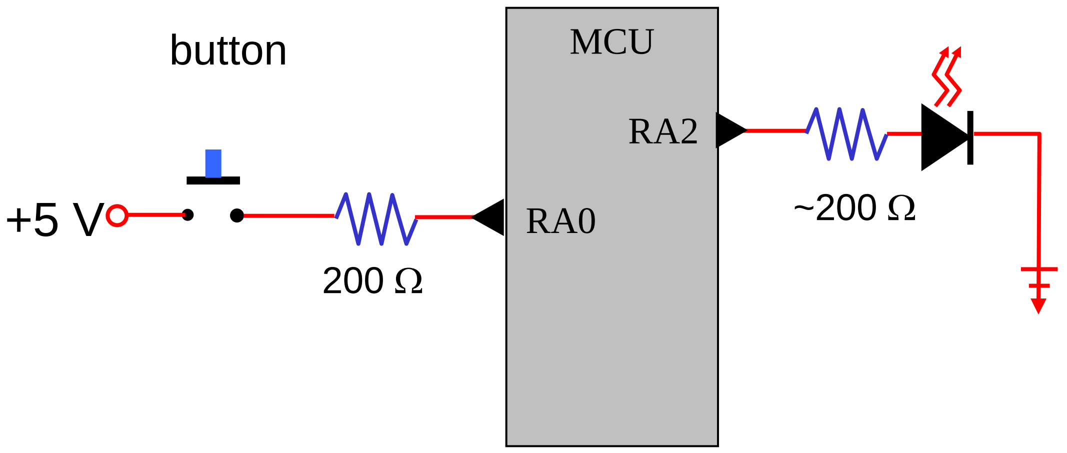
Voltage divider $V_B = V_{DD}\dfrac{R_{input}}{200\Omega + R_{input}} \cong V_{DD}$

This circuit keeps the input current low but why doesn't it work?



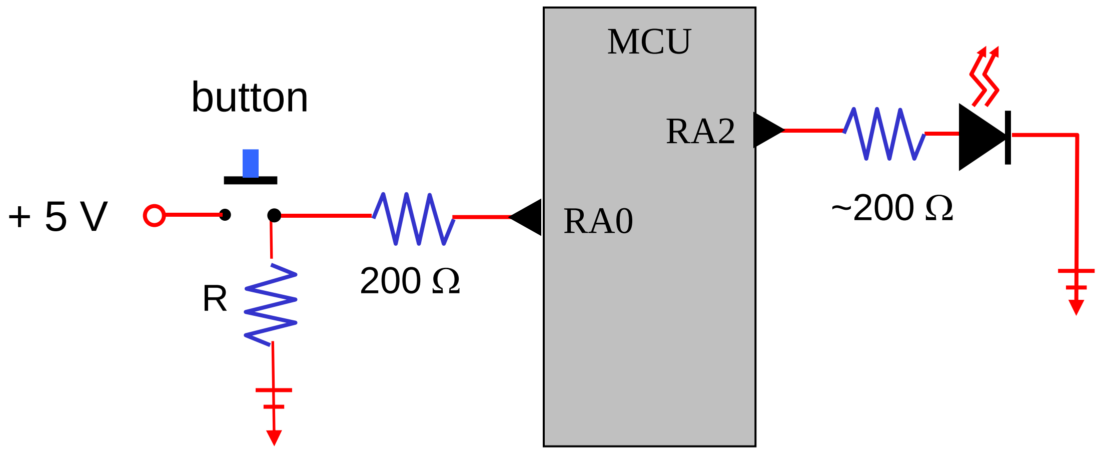
A. Voltage at RA0 is ~2.5 V

This circuit keeps the input current low and has the right voltage at the pin when the button is down. Why doesn't it work?



A. When the button is up, RA0 is floating. Random noise can set it high.

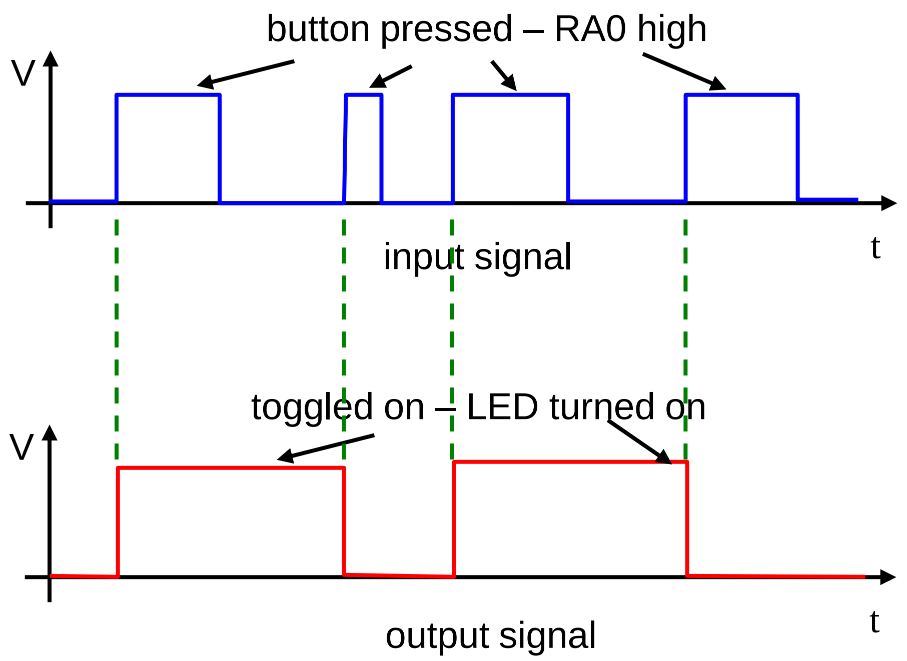
Added a resistor keeps RA0 grounded when button is up. When button is pressed, do have a current through R. Need R big to keep current low.



Toggles

- Sample code turned LED on only when button is down.
- What is we want next is a toggle. Press once and LED is on, press again and LED if off.
- Need to analyze signal and write code for a toggle.

What are we looking for?



Rising edges (0V → 5V) indicates when to change.

```
#include "..\Functions\buttons.h"

void switch1_risingedge_action(void);
void switch1_fallingedge_action(void);

int main(void)
{
    unsigned char has_switch1_changed = 0;
    TRISDbits.RD1 = 1; // set RD1 as input (switch 1)

    while(1)
    {
      has_switch1_changed =
              monitor_switch1_for_edges(PORTDbits.RD1);
      if ( has_switch1_changed == 1 )
              switch1_risingedge_action();
      if ( has_switch1_changed == 2 )
              switch1_fallingedge_action();
    }
}
```

```
void switch1_risingedge_action(void)
{
    // do something useful or leave blank
}

void switch1_fallingedge_action(void)
{
    // do something useful or leave blank
}
```

```
// buttons.c
#include "buttons.h"

unsigned char last_switch1_value = 0;

// returns 0 if no edge has happened
// returns 1 at a rising edge
// returns 2 at a falling edge

unsigned char monitor_switch1_for_edges(unsigned char
digitalinputpin)
{
    unsigned char has_switch1_changed = 0;

    if (last_switch1_value == 0 && digitalinputpin)
    { last_switch1_value = 1;
      has_switch1_changed = 1;}

    if (last_switch1_edge == 1 && !digitalinputpin)
    { last_switch1_value = 0;
      has_switch1_changed = 2;}

    return has_switch1_changed;
}
```