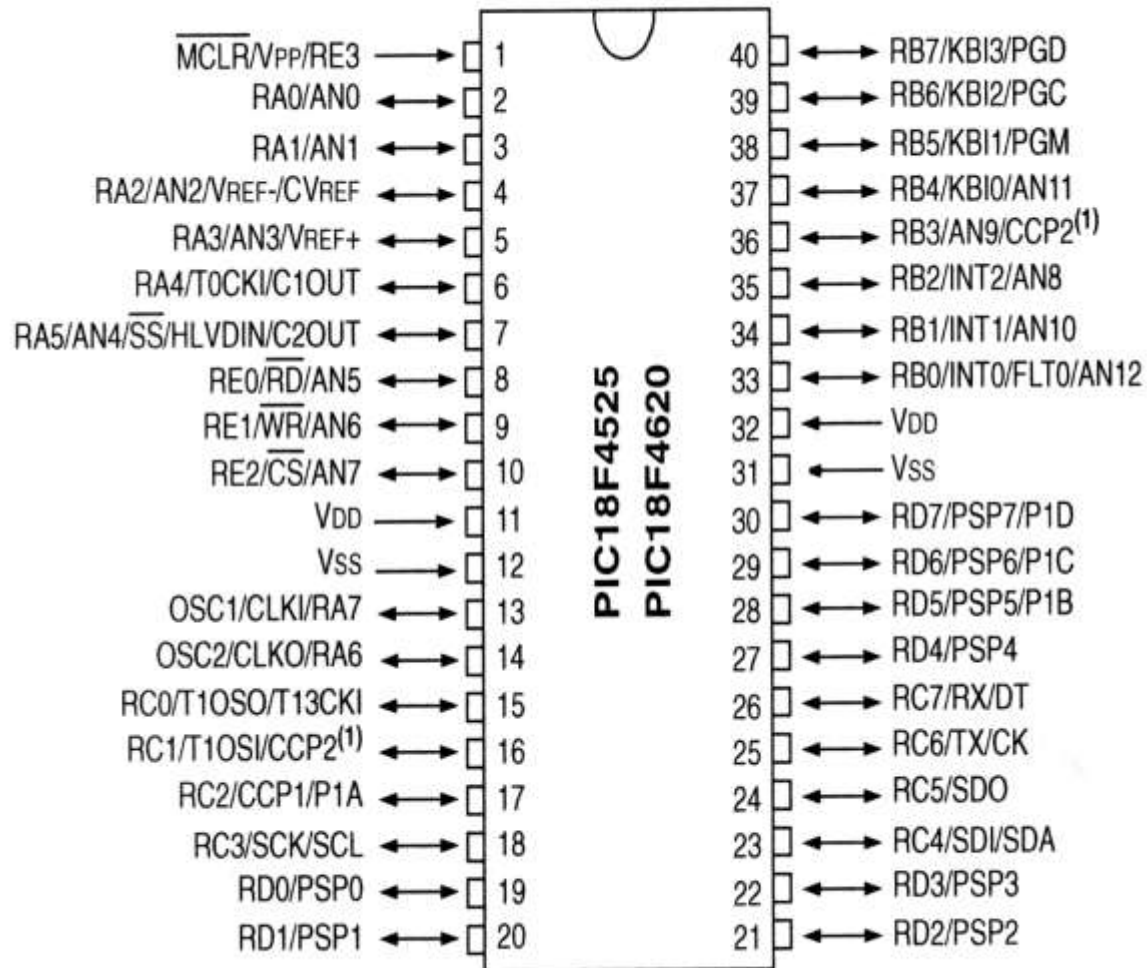


DIO

- 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.



RA_n, RB_n, RC_n, RD_n $n = 0$ to 7

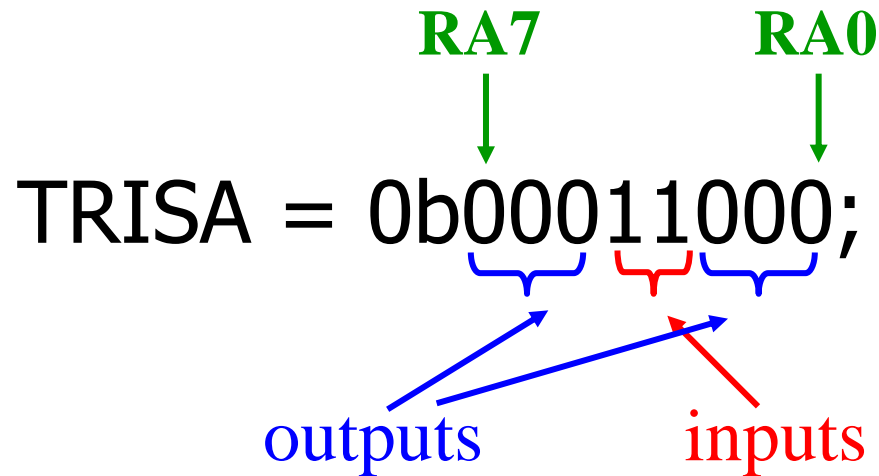
DIO SFRs

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



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.

high low
↓ ↓
PORTA = 0b11101000;

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.TRISXn` and `PORTXbits.RXn` address bit level

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

```
void main(void)
```

```
{
```

```
    TRISAbits.TRISA3 = 1; // set RA3 as input
```

```
    TRISAbits.TRISA2 = 0; // set RA2 as output
```

```
    TRISAbits.TRISA1 = 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.TRISC2 = 0;  
    TRISCbits.TRISC4 = 0;  
    TRISBbits.TRISB0 = 1;  
    TRISBbits.TRISB3 = 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);
```

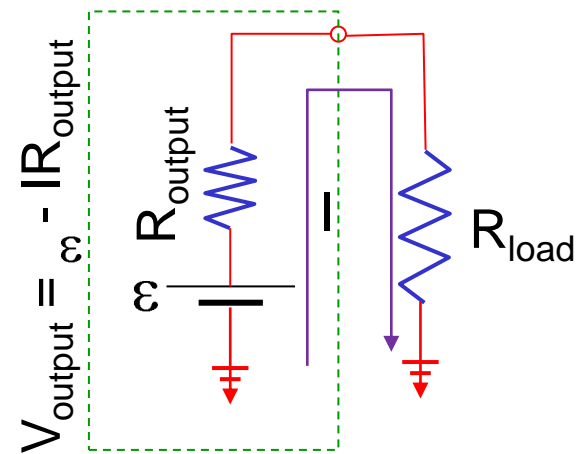
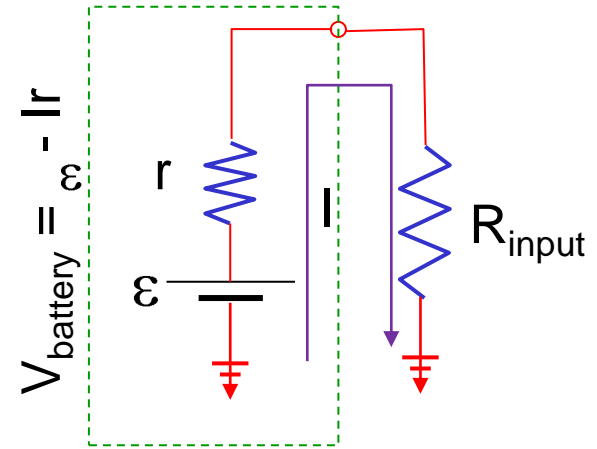
```
}
```

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 $\sim 5 \text{ K}\Omega$ to act as a voltmeter

- An output pin has/needs a small resistance $\sim 50 \Omega$ to act as a “battery”



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.

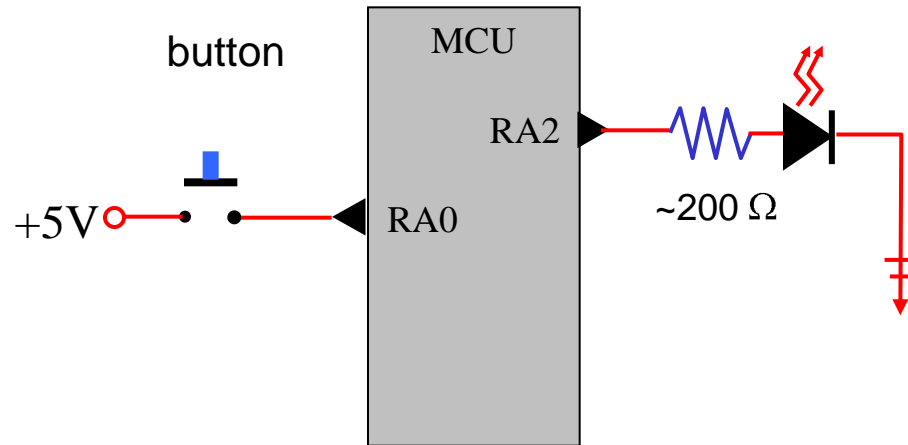
Ohm's law: $5V/25 \text{ mA} = 200 \Omega$

\therefore Connect at least $R_{\text{load}} \sim 200 \Omega$ 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

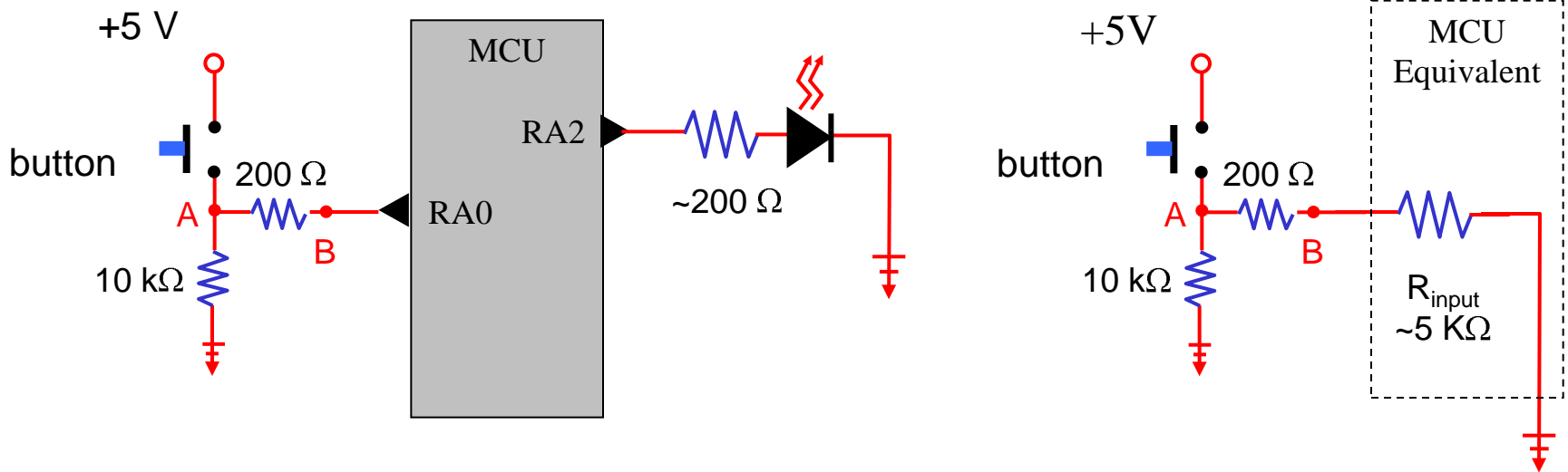
The Wrong Way!!!



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

Use a “logic” switch instead.

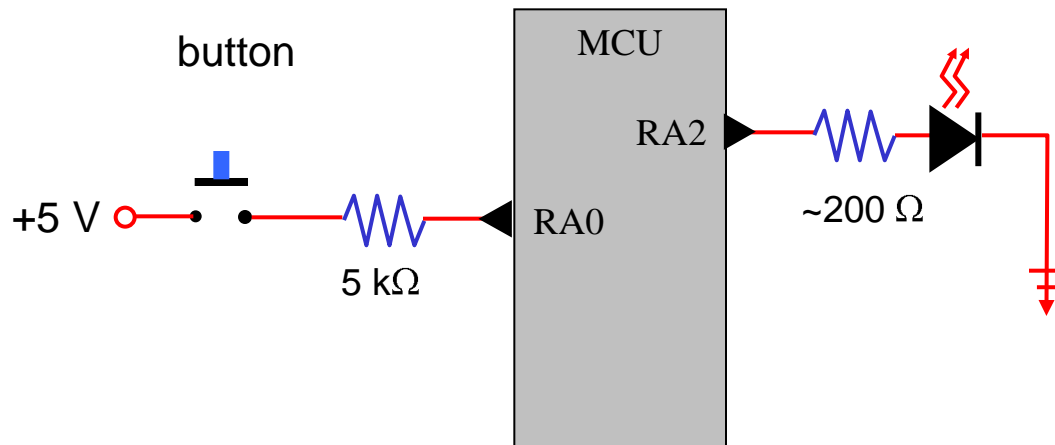
The Right Way



Input has a large resistance.

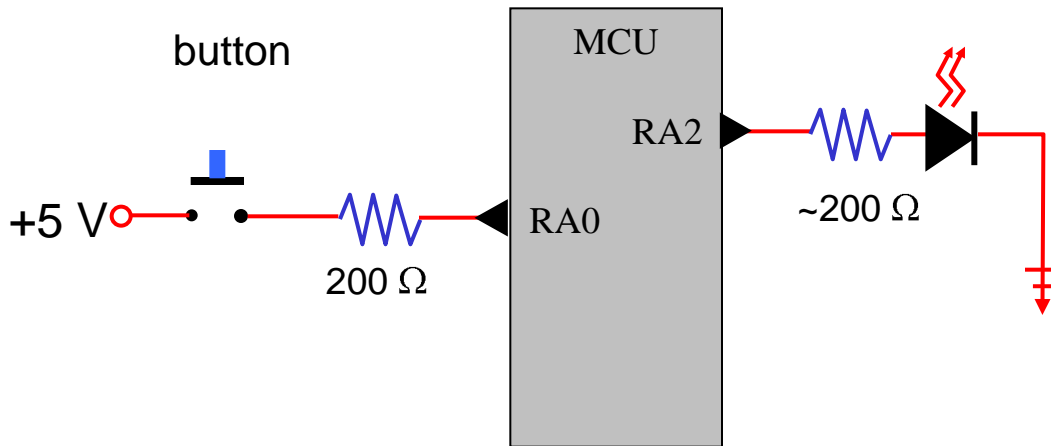
$$\text{Voltage divider } V_B = V_{DD} \frac{R_{input}}{200\Omega + R_{input}} \cong V_{DD}$$

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



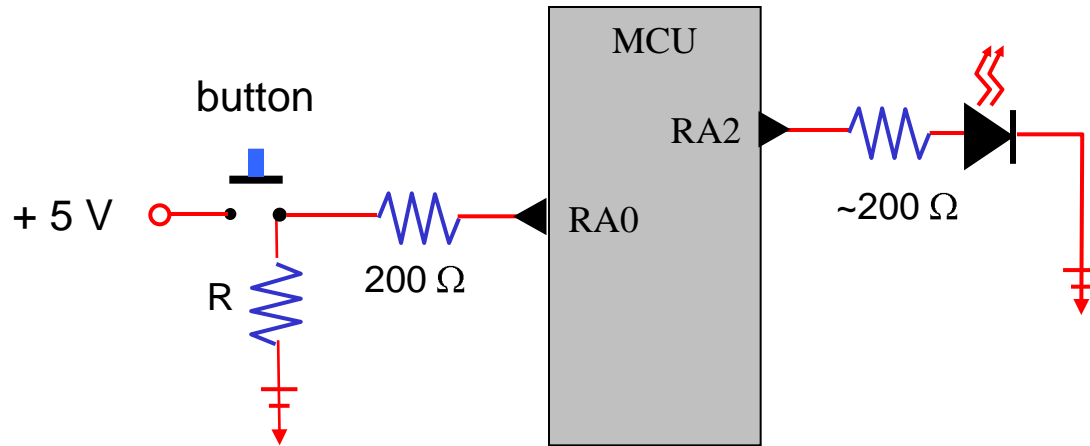
A. Voltage at RA0 is ~ 2.5 V

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.

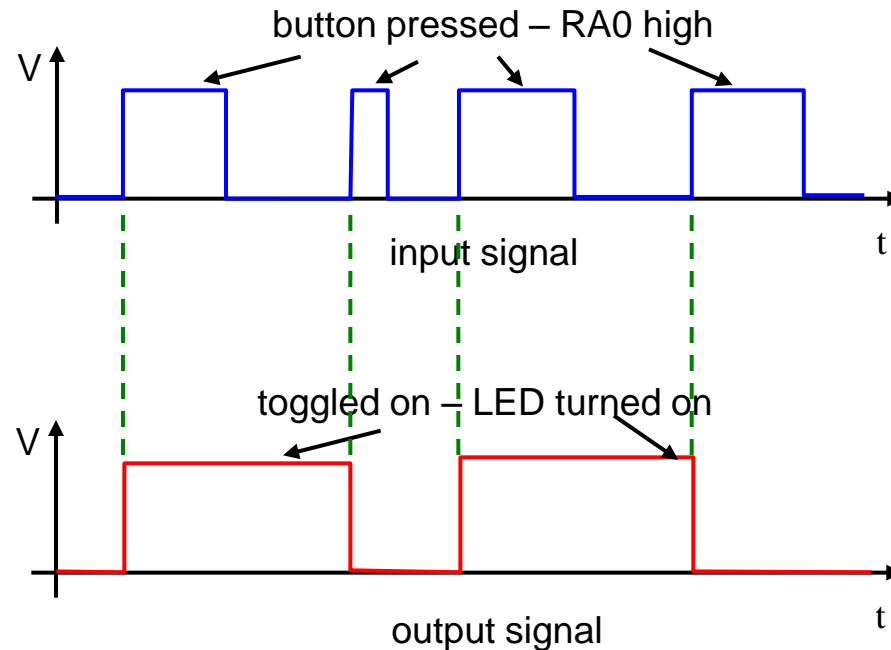
Added a resistor keeps RA0 grounded when button is up. When button is pressed, do have a current through R. Need R big to keep current low.



Toggles

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

What are we looking for?



Rising edges ($0V \rightarrow 5V$) indicates when to change.

```
void monitor_switch1_for_edges(void);
void switch1_risingedge_action(void);

char last_switch1_edge = 0; // last edge, start with low
                           // global variable

void main(void)
{
    TRISAbits.TRISA0 = 1; // set RA0 as input (switch 1)
    while(1)
    {
        monitor_switch1_for_edges();
    }
}
```

```
void monitor_switch1_for_edges(void)
{
    if (last_switch1_edge == 0 && PORTAbits.RA0 )
        { // rising edge detected if PORTAbits.RA0 is 1 (on)
            switch1_risingedge_action();
            last_switch1_edge = 1; // rising switch edge detected
        }
    if (last_switch1_edge == 1 && !PORTAbits.RA0)
        { // falling edge detected if PORTAbits.RA0 is 0 (off)
            last_switch1_edge = 0; // falling switch edge detected
        }
}
```

```
void switch1_risingedge_action(void)
{
    // do something useful
}
```

```
#define OFF 0
#define ON 1

void main(void){
unsigned char old_RA0_input = 0, new_RA0_input, switch = OFF;
configure RA0 as input;

while(1){
    new_RA0_input = PORTAbits.RA0;
    if( (old_RA0_input == 0) && (new_RA0_input == 1) ){
        // rising edge found
        switch = !switch; // flip switch
        if (switch == ON){
            do something;
        }
        else{
            do opposite something;
        } //end if
    } // end if
} // end while
} // end main
```

```
#define OFF 0
#define ON 1

void main(void){
unsigned char old_RA0_input = 0, new_RA0_input, switch = OFF;
configure RA0 as input;

while(1){
    new_RA0_input = PORTAbits.RA0;
    if( !old_RA0_input && new_RA0_input ){
        // if 0 and 1, rising edge found
        switch = !switch; // flip switch
        if (switch){ // 1 = ON

            do something;
        }
        else{
            do opposite something;
        } //end if
    } // end if
} // end while
} // end main
```

```

#define NO 0
#define YES 1

unsigned char switch_changed = NO; // global variable
unsigned char RisingEdgeDetector(unsigned char old_RA0_input, unsigned char new_RA0_input);

void main(void){
    unsigned char old_RA0_input = 0;
    configure RA0 as input;

    while(1){
        old_RA0_input = RisingEdgeDetector(old_RA0_input, PORTAbits.RA0);
        if (switch_changed){
            do something;
        }
        else{
            do opposite something;
        } //end if else
    } // end while
} // end main

unsigned char RisingEdgeDetector(unsigned char old_RA0_input, unsigned char new_RA0_input){
    if( !old_RA0_input && new_RA0_input ){
        // if 0 and 1, rising edge found
        switch_changed = YES;
        old_RA0_input = new_RA0_input;
    }
    else{
        switch_changed = NO;
    }
    return old_RA0_input;
}

```