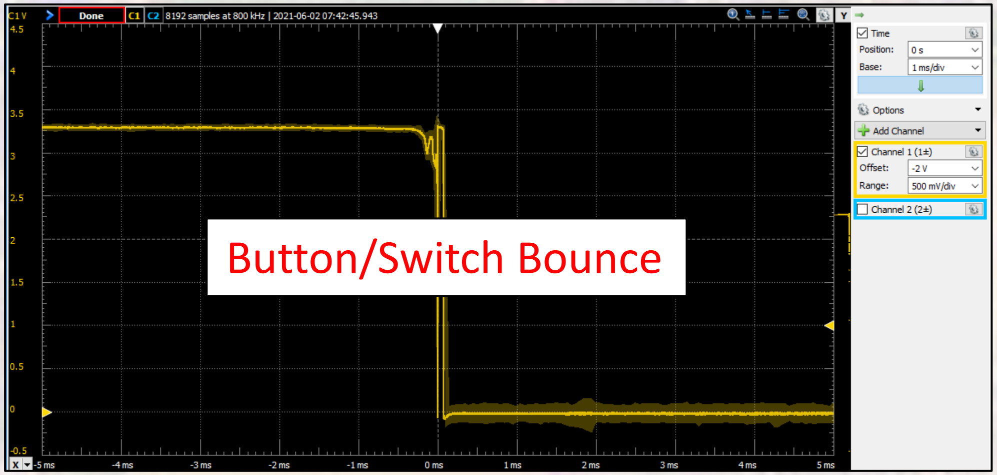
Debounce

Last updated – 7/4/21

Debounce

- When a button is pressed (or released) it often bounces
    - This causes the pin associated with the button to oscillate between 0 and 1



8192 samples at 800 kHz | 2021-06-02 07:42:45.943

☑ Time
Position: 0 s
Base: 1 ms/div

☑ Channel 1 (1±)
Offset: -2 V
Range: 500 mV/div

☐ Channel 2 (2±)

Button/Switch Bounce

- Under normal polling this is not an issue (why?)
- When using pin-interrupts this can cause multiple interrupts and un-intended “presses” or “releases”

Debounce

- This problem is very complex
  - There are hardware and software solutions
  - Hardware solutions can be made very robust – but may not be practical (or available) on our board
  - Software solutions are not 100% effective
  - Any solution we choose has some failure mechanism

Note: typically the bouncing is resolved in less than a few milli-seconds

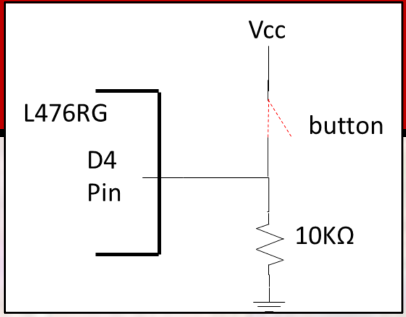
Debounce

- Simple software based debounce solution

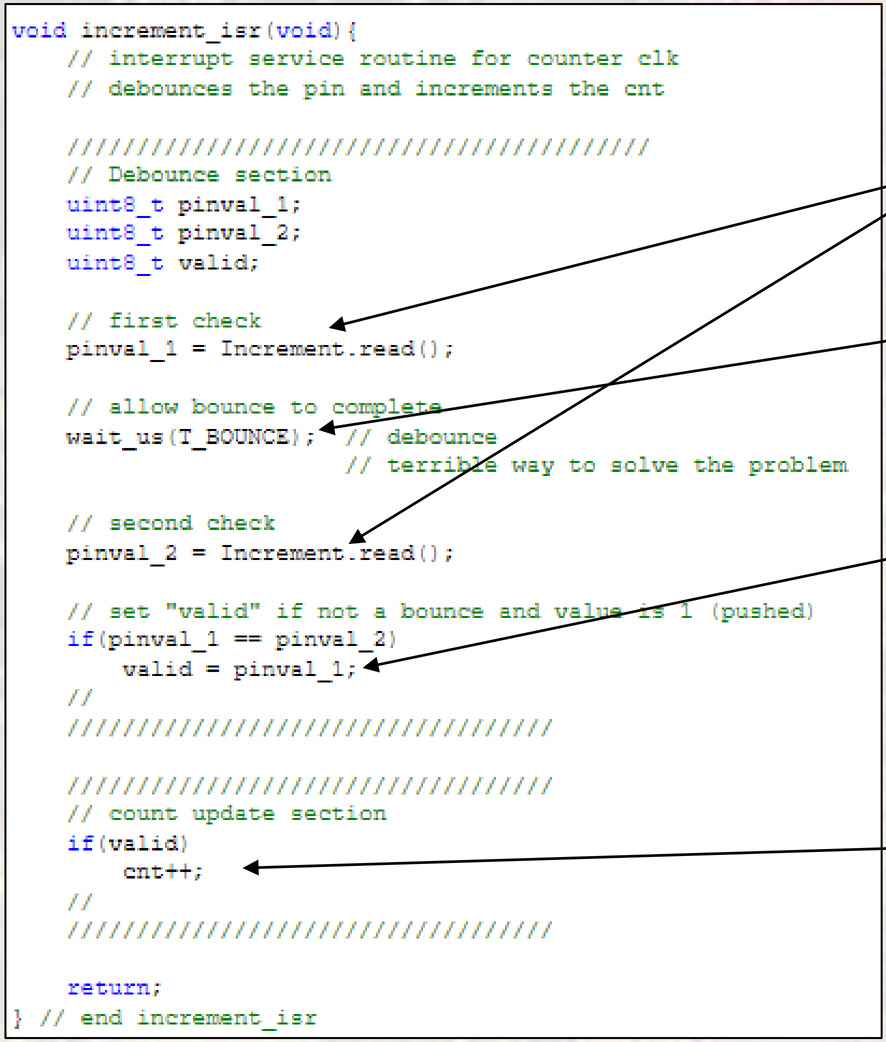
  - We can check the pin, wait a few milli-seconds and check again
    - If the pin is different we may be bouncing – do not update the value
    - If the pin is the same we know we are not bouncing – “valid”

  - Update the value with the new “valid” pin value

Debounce



- Debounce section of an ISR - example



```
void increment_isr(void){
    // interrupt service routine for counter clk
    // debounces the pin and increments the cnt

    //////////////////////////////////////////
    // Debounce section
    uint8_t pinval_1;
    uint8_t pinval_2;
    uint8_t valid;

    // first check
    pinval_1 = Increment.read();

    // allow bounce to complete
    wait_us(T_BOUNCE);    // debounce
                          // terrible way to solve the problem

    // second check
    pinval_2 = Increment.read();

    // set "valid" if not a bounce and value is 1 (pushed)
    if(pinval_1 == pinval_2)
        valid = pinval_1;
    //
    ///////////////////////////////////

    ///////////////////////////////////
    // count update section
    if(valid)
        cnt++;
    //
    ///////////////////////////////////

    return;
} // end increment_isr
```

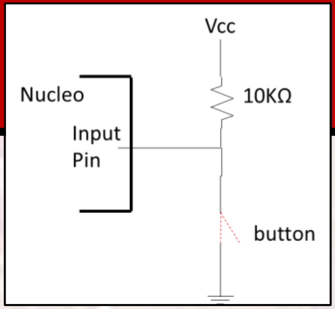
InterruptIn called Increment
Set for a rise edge to initiate the ISR

#define for bounce delay (5000) us

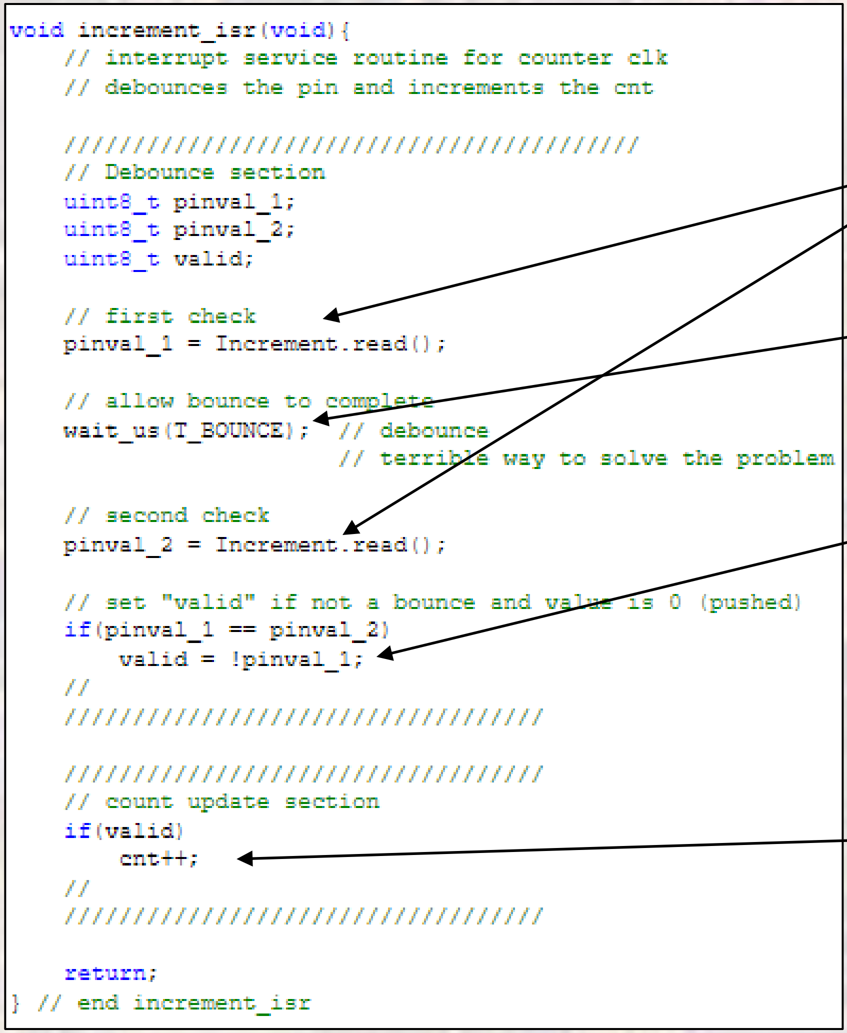
Valid set to 1 if button pressed
Valid set to 0 if button not pressed (released)

In this case incrementing the cnt variable on a button press – this is the section that would change depending on the need

Debounce



- Debounce section of an ISR - example



```
void increment_isr(void){
    // interrupt service routine for counter clk
    // debounces the pin and increments the cnt

    //////////////////////////////////////////
    // Debounce section
    uint8_t pinval_1;
    uint8_t pinval_2;
    uint8_t valid;

    // first check
    pinval_1 = Increment.read();

    // allow bounce to complete
    wait_us(T_BOUNCE);  // debounce
                        // terrible way to solve the problem

    // second check
    pinval_2 = Increment.read();

    // set "valid" if not a bounce and value is 0 (pushed)
    if(pinval_1 == pinval_2)
        valid = !pinval_1;
    //
    ///////////////////////////////////

    ///////////////////////////////////
    // count update section
    if(valid)
        cnt++;
    //
    ///////////////////////////////////

    return;
} // end increment_isr
```

InterruptIn called Increment
Set for a fall edge to initiate the ISR

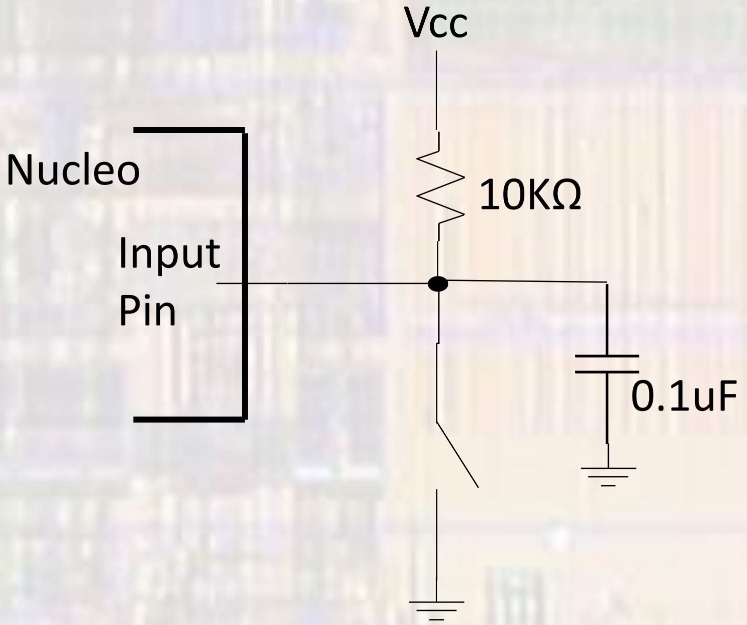
#define for bounce delay (5000) us

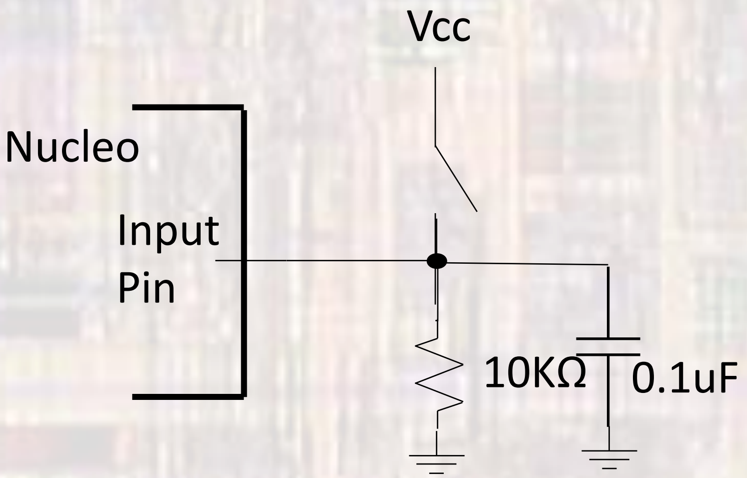
Valid set to 1 if button pressed (!)
Valid set to 0 if button not pressed (released)

In this case incrementing the cnt variable on a button press – this is the section that would change depending on the need

Debounce

- Basic hardware solution - Adding a capacitor
  - Slows down the transition and prevents bouncing
  - Adds the cost of the capacitor





Assume 5ms for debouncing AND 4 or 5 time constants to transition → $\tau = RC = 1\text{ms}$

$C = \tau/R$ → $C = 1\text{ms}/10\text{K}\Omega = 0.1\text{uF} = 100\text{nF} = 100{,}000\text{pF}$ → 104 marking