

EEL 4742 – Embedded Systems

Module 11 – Advanced Interrupts

Hadi Kamali

Department of Electrical and Computer Engineering (ECE)
University of Central Florida

*Office Location/phone: HEC435 – (407) 823-0764
webpage: <https://www.ece.ucf.edu/~kamali/>
e-mail: kamali@ucf.edu*

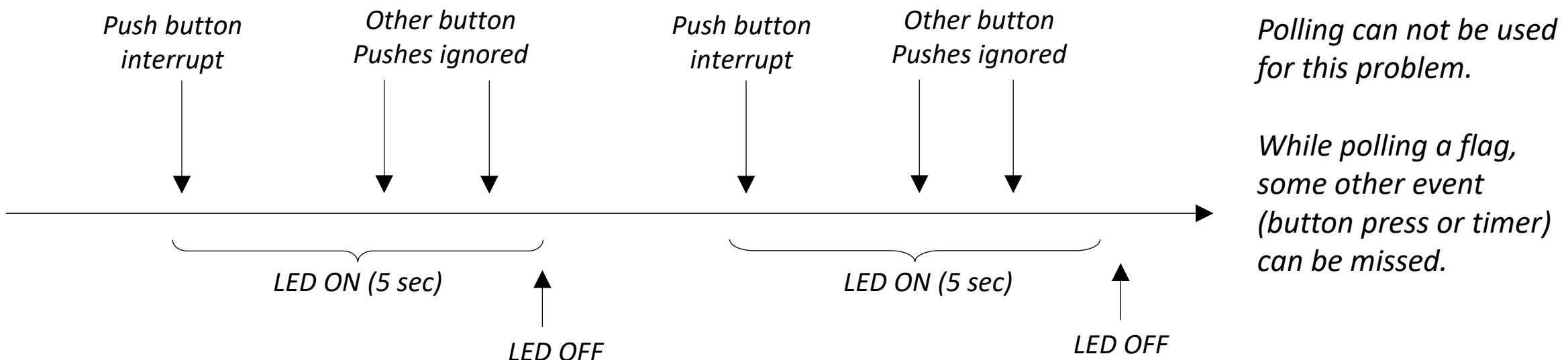
HAVEN Research Group
<https://haven.ece.ucf.edu/>



Single Events (Sequential)

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - But ignore the additional button presses when the LED is ON.



Single Events (Sequential)

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - But ignore the additional button presses when the LED is ON.

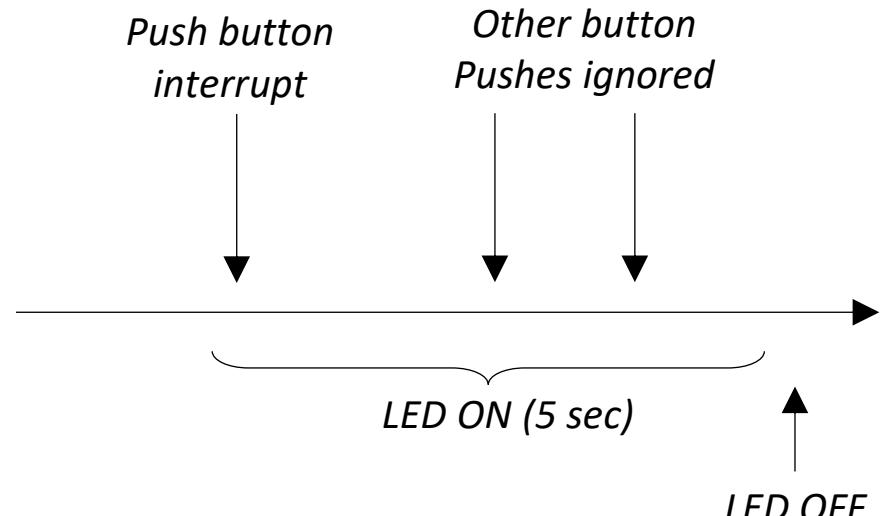
<i>Event</i>	<i>Actions to be performed</i>
Button pressed	<ul style="list-style-type: none"> Turn ON LED <u>Disable</u> external interrupt Enable timer
Timer completes 5 second	<ul style="list-style-type: none"> Turn OFF LED <u>Enable</u> external interrupt Disable timer

To turn ON/OFF LED

To ignore further button presses

To provide a 5 second delay using timer

A breakdown of the events that happen and what actions are to be performed at each event.





Single Events (Sequential)

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.

- But ignore the additional button presses when the LED is ON.

Event	Actions to be performed
Button pressed	<ul style="list-style-type: none">• Turn ON LED• <u>Disable</u> external interrupt• Enable timer
Timer completes 5 second	<ul style="list-style-type: none">• Turn OFF LED• <u>Enable</u> external interrupt• Disable timer

```
// Code for these events
#include <msp430fr6989.h>
// Define BITi for LED(s)
void main(void)
{
    // stop watchdog timer
    WDTCLTC = WDTPW | WDTHOLD;

    // Configure LED (output) and push button (input)
    P1DIR |= redLED;      // Direct pin as output
    P1SEL1 &= ~LED;        // Primary function, P1.0
    P1OUT &= ~redLED;     // Turn LED Off

    // Enable external interrupts
    P1IE |= BUTTON;
    P1IES |= BUTTON;
    P1IFG &= ~BUTTON;

    // Clear TA0CTL register
    _low_power_mode_3(); // Enter low power mode 3
    return;
}
```

```
#pragma vector = PORT1_VECTOR
__interrupt void port1_ISR(void)
{
    // clear the interrupt flag
    ...

    // Turn on LED
    ...

    // Disable external interrupt
    P1IE &= ~BUTTON;

    // Configure timer for 5 seconds
    TA0CCR0 = ???;
    TA0CTL |= TASSEL_1|ID_2|MC_1|TACLR|TAIE;
    TA0CTL &= ~TAIFG;
}
```

why?



Single Events (Sequential)

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - But ignore the additional button presses when the LED is ON.

Event	Actions to be performed
Button pressed	<ul style="list-style-type: none"> Turn ON LED <u>Disable</u> external interrupt Enable timer
Timer completes 5 second	<ul style="list-style-type: none"> Turn OFF LED <u>Enable</u> external interrupt Disable timer

```

// Code for these events
#include <msp430fr6989.h>
// Define BITi for LED(s)
void main(void)
{
    // stop watchdog timer
    WDTCLTC = WDTPW | WDTHOLD;

    // Configure LED (output) and push button (input)
    P1DIR |= redLED;      // Direct pin as output
    P1SEL1 &= ~LED;        // Primary function, P1.0
    P1OUT &= ~redLED;      // Turn LED Off

    // Enable external interrupts
    P1IE |= BUTTON;
    P1IES |= BUTTON;
    P1IFG &= ~BUTTON;

    // Clear TA0CTL register
    _low_power_mode_3(); // Enter low power mode 3
    return;
}

```

```

#pragma vector = PORT1_VECTOR
__interrupt void port1_ISR(void)
{
    // clear the interrupt flag
    ...

    // Turn on LED
    ...

    // Disable external interrupt
    P1IE &= ~BUTTON;

    // Configure timer for 5 seconds
    TA0CCR0 = ???;
    TA0CTL |= TASSEL_1|ID_2|MC_1|TACLR|TAIE;
    TA0CTL &= ~TAIFG;
}

```

With max value (0xFFFF), it is 2 seconds!

Divider is used with up mode for 5 seconds!



Single Events (Sequential)

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.

- But ignore the additional button presses when the LED is ON.

```
#pragma vector = TIMER0_A1_VECTOR
__interrupt void TOA1_ISR(void)
{
    // clear the interrupt flag
    TA0CTL &= ~TAIFG;

    // Turn off LED
    ...

    // Enable external interrupt
    P1IE |= BUTTON;
    P1IE &= ~P1IFG;

    // Disable timer interrupt
    TA0CTL &= ~TAIE;
}
```

```
// Code for these events
#include <msp430fr6989.h>
// Define BITi for LED(s)
void main(void)
{
    // stop watchdog timer
    WDTCLTC = WDTPW | WDTHOLD;

    // Configure LED (output) and push button (input)
    P1DIR |= redLED;      // Direct pin as output
    P1SEL1 &= ~LED;        // Primary function, P1.0
    P1OUT &= ~redLED;      // Turn LED Off

    // Enable external interrupts
    P1IE |= BUTTON;
    P1IES |= BUTTON;
    P1IFG &= ~BUTTON;

    // Clear TA0CTL register
    _low_power_mode_3(); // Enter low power mode 3
    return;
}
```

```
#pragma vector = PORT1_VECTOR
__interrupt void port1_ISR(void)
{
    // clear the interrupt flag
    ...

    // Turn on LED
    ...

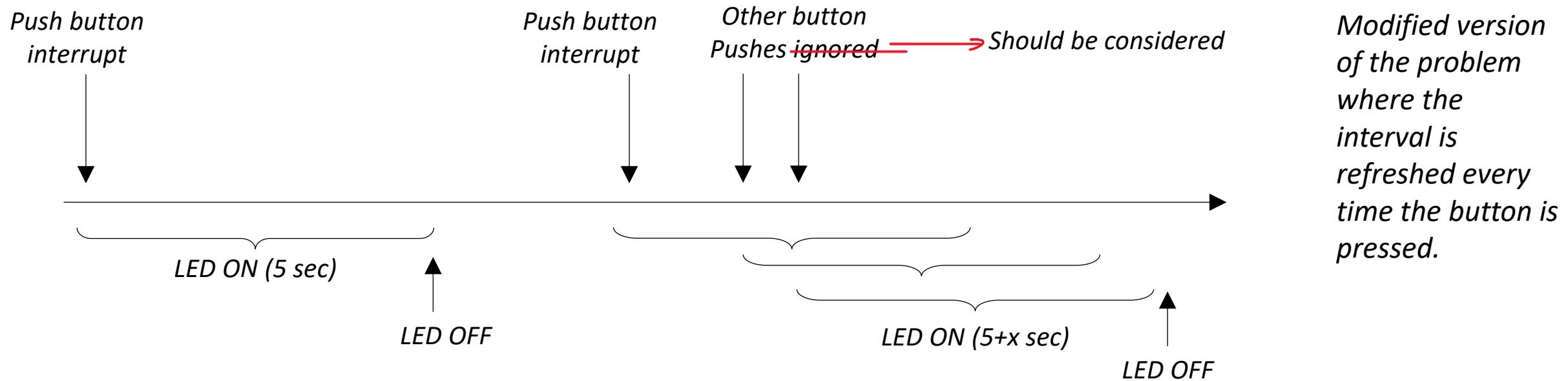
    // Disable external interrupt
    P1IE &= ~BUTTON;

    // Configure timer for 5 seconds
    TA0CCR0 = ???;
    TA0CTL |= TASSEL_1|ID_2|MC_1|TACLR|TAIE;
    TA0CTL &= ~TAIFG;
}
```

Timer is also based on ISR!

Concurrent Events (No Waiting)

- Example: The use of push button
 - Problem 2: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - If the button is pressed again before the 5 seconds, then restart the 5 seconds from that moment.

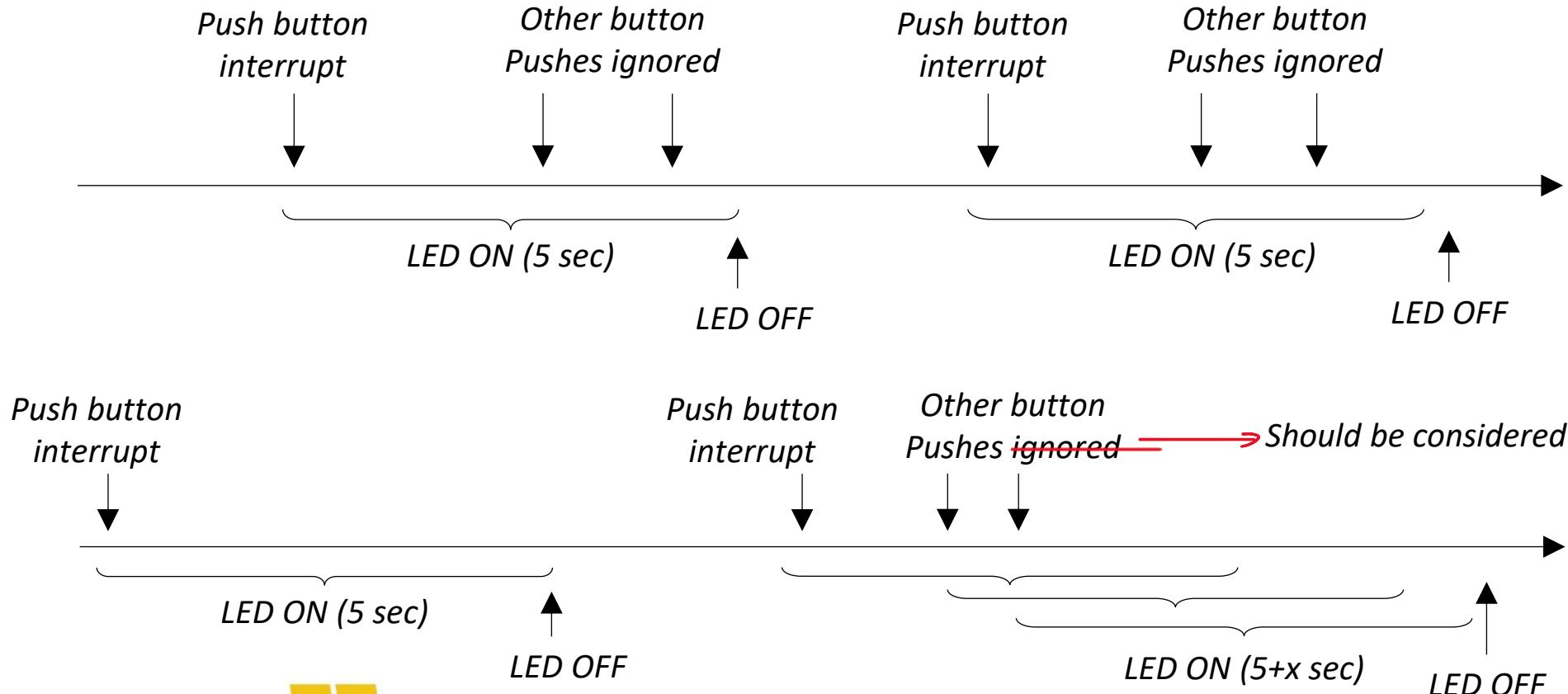


Concurrent Events (No Waiting)

- Example: The use of push button

- Problem 2: Every time the push button is pressed, turn ON the LED for 5 seconds.

- If the button is pressed again before the 5 seconds, then restart the 5 seconds from that moment.



Modified version
of the problem
where the
interval is
refreshed every
time the button is
pressed.

Single Events (Sequential)

- Example: The use of push button

- Problem 2: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - If the button is pressed again before the 5 seconds, then restart the 5 seconds from that moment.

Event	<i>Actions to be performed</i>
Button pressed	<ul style="list-style-type: none"> Turn ON LED <u>Disable</u> external interrupt Enable timer
Timer completes 5 second	<ul style="list-style-type: none"> Turn OFF LED <u>Enable</u> external interrupt Disable timer

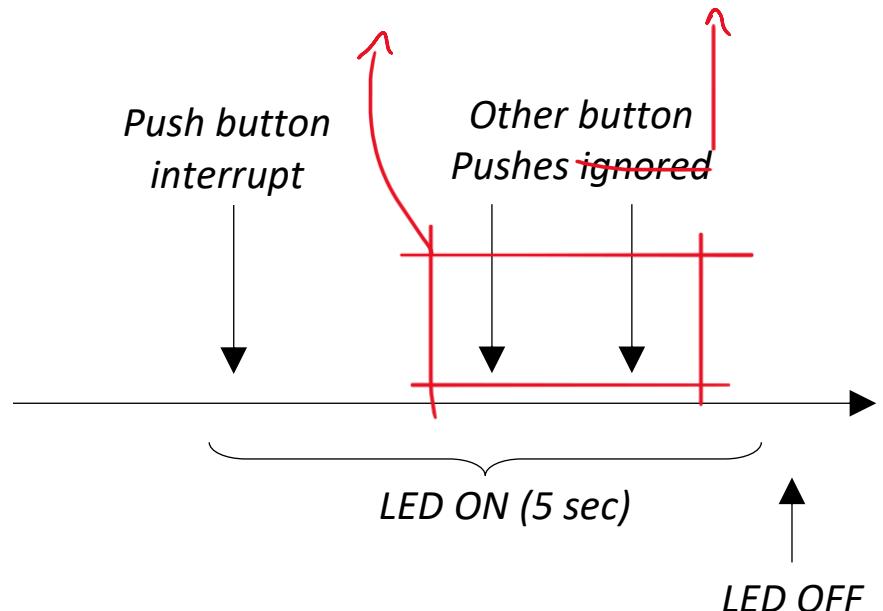
To turn ON/OFF LED

To ignore further button presses

To provide a 5 second delay using timer

A breakdown of the events that happen and what actions are to be performed at each event.

Should be considered!
There is not need to disable the external interrupts!



Concurrent Events (No Waiting)

- Example: The use of push button

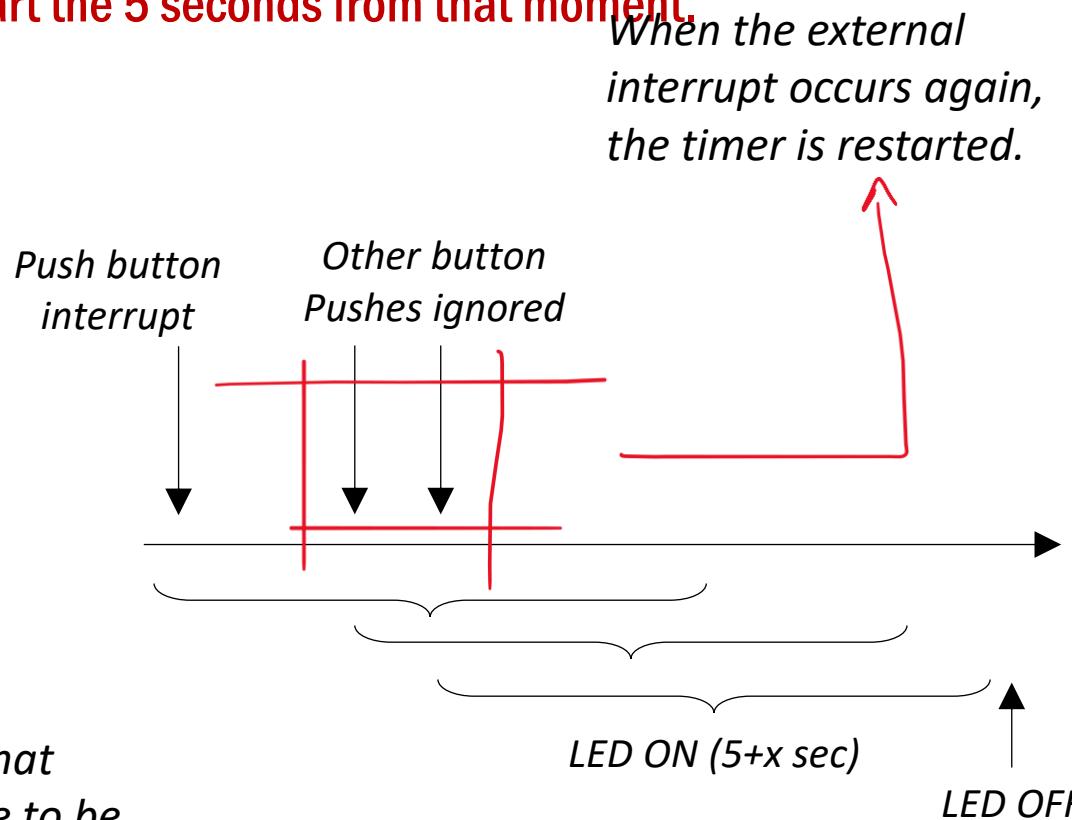
- Problem 2: Every time the push button is pressed, turn ON the LED for 5 seconds.

- If the button is pressed again before the 5 seconds, then restart the 5 seconds from that moment.

Event	Actions to be performed
Button pressed	<ul style="list-style-type: none"> Turn ON LED <u>Disable</u> external interrupt Enable timer
Timer completes 5 second	<ul style="list-style-type: none"> Turn OFF LED <u>Enable</u> external interrupt Disable timer

To turn ON/OFF LED
 To ignore further button presses
 To provide a 5 second delay using timer

A breakdown of the events that happen and what actions are to be performed at each event.





Concurrent Events (No Waiting)

- Example: The use of push button

- Problem 2: Every time the push button is pressed, turn ON the LED for 5 seconds.

- If the button is pressed again before the 5 seconds, then restart the 5 seconds from that moment.

```
#pragma vector = TIMER0_A1_VECTOR
__interrupt void TOA1_ISR(void)
{
    // clear the interrupt flag
    TA0CTL &= ~TAIFG;

    // Turn off LED
    ...

    // Enable external interrupt
    P1IE |= BUTTON;
    P1IE &= ~P1IFG;

    // Disable timer interrupt
    TA0CTL &= ~TAIE;
}
```

No need to
disable it!

```
// Code for these events
#include <msp430fr6989.h>
// Define BITi for LED(s)
void main(void)
{
    // stop watchdog timer
    WDTCLTC = WDTPW | WDTHOLD;

    // Configure LED (output) and push button (input)
    P1DIR |= redLED;      // Direct pin as output
    P1SEL1 &= ~LED;        // Primary function, P1.0
    P1OUT &= ~redLED;      // Turn LED Off

    // Enable external interrupts
    P1IE |= BUTTON;
    P1IES |= BUTTON;
    P1IFG &= ~BUTTON;

    // Clear TA0CTL register
    _low_power_mode_3(); // Enter low power mode 3
    return;
}
```

```
#pragma vector = PORT1_VECTOR
__interrupt void port1_ISR(void)
{
    // clear the interrupt flag
    ...
    // Turn on LED
    ...
    // Disable external interrupt
    P1IE &= ~BUTTON;

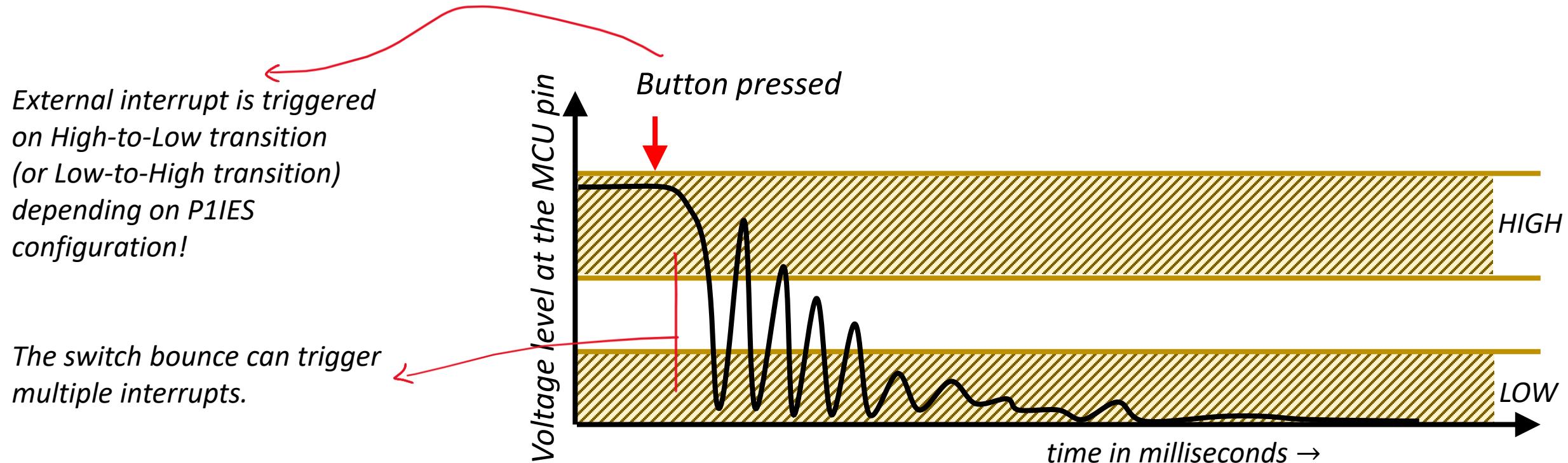
    // Configure timer for 5 seconds
    TA0CCR0 = ???;
    TA0CTL |= TASSEL_1|ID_2|MC_1|TACLR|TAIE;
    TA0CTL &= ~TAIFG;
}
```

No need to disable it!

Timer is also based on ISR!

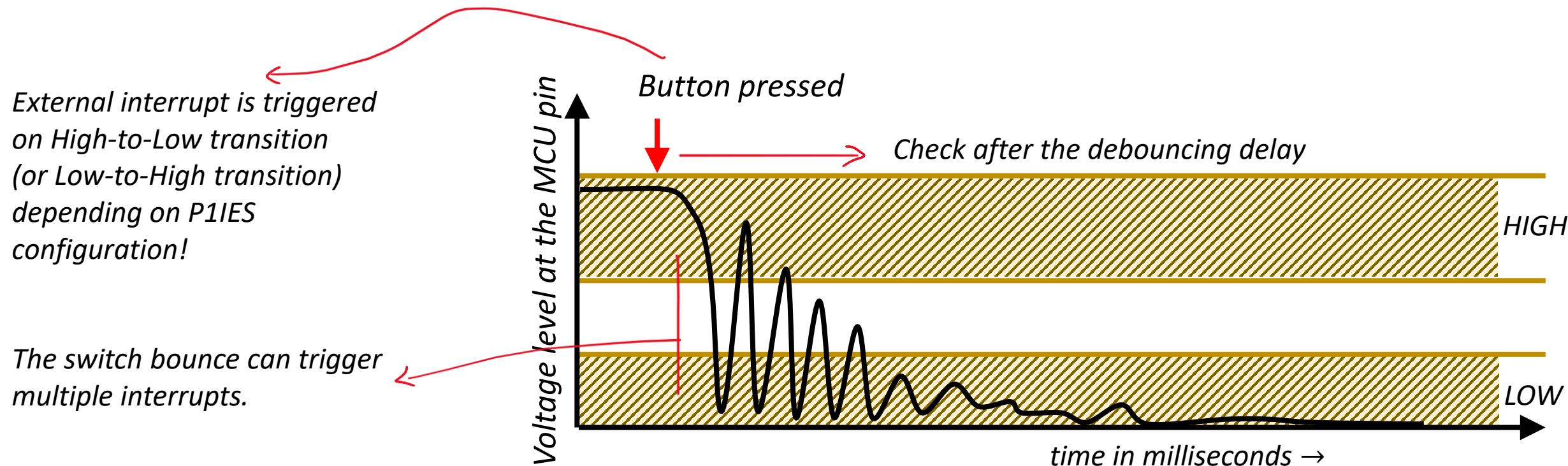
Switch Bouncing Problem

- Multiple Interrupts for a physical-oriented events
 - Noise of the system



Concurrent Interrupt for Switch Bouncing Problem

- Multiple Interrupts for a physical-oriented events
 - Noise of the system
 - Every time the push button is pressed, start a timer for 20 ms and ignore additional button presses.
 - After the 20 ms, re-check whether the button is still pushed. If it is pushed, then consider as input.



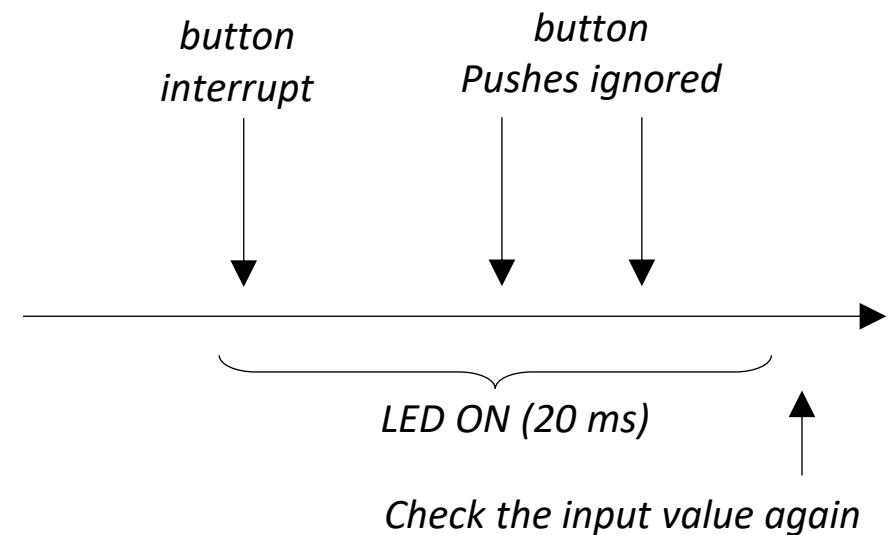
Checking for Bouncing

- Example: The use of push button

- Problem 1: Every time the push button is pressed, turn ON the LED for 5 seconds.
 - But ignore the additional button presses when the LED is ON.

<i>Event</i>	<i>Actions to be performed</i>
<i>Button pressed</i>	<ul style="list-style-type: none"> Disable external interrupt Enable timer
<i>Timer completes 20 ms</i>	<ul style="list-style-type: none"> Enable external interrupt Disable timer Check input again

To provide a 20 ms delay using timer



Things to consider while debouncing

- Can a rapid button press (e.g. gaming) be sensed accurately?
 - if debouncing takes 50 ms each time?
- What happens if the button is pressed at the rate of 20 clicks per second?
- Quality of the switches affect the debouncing period.



Toggle switch



Push button

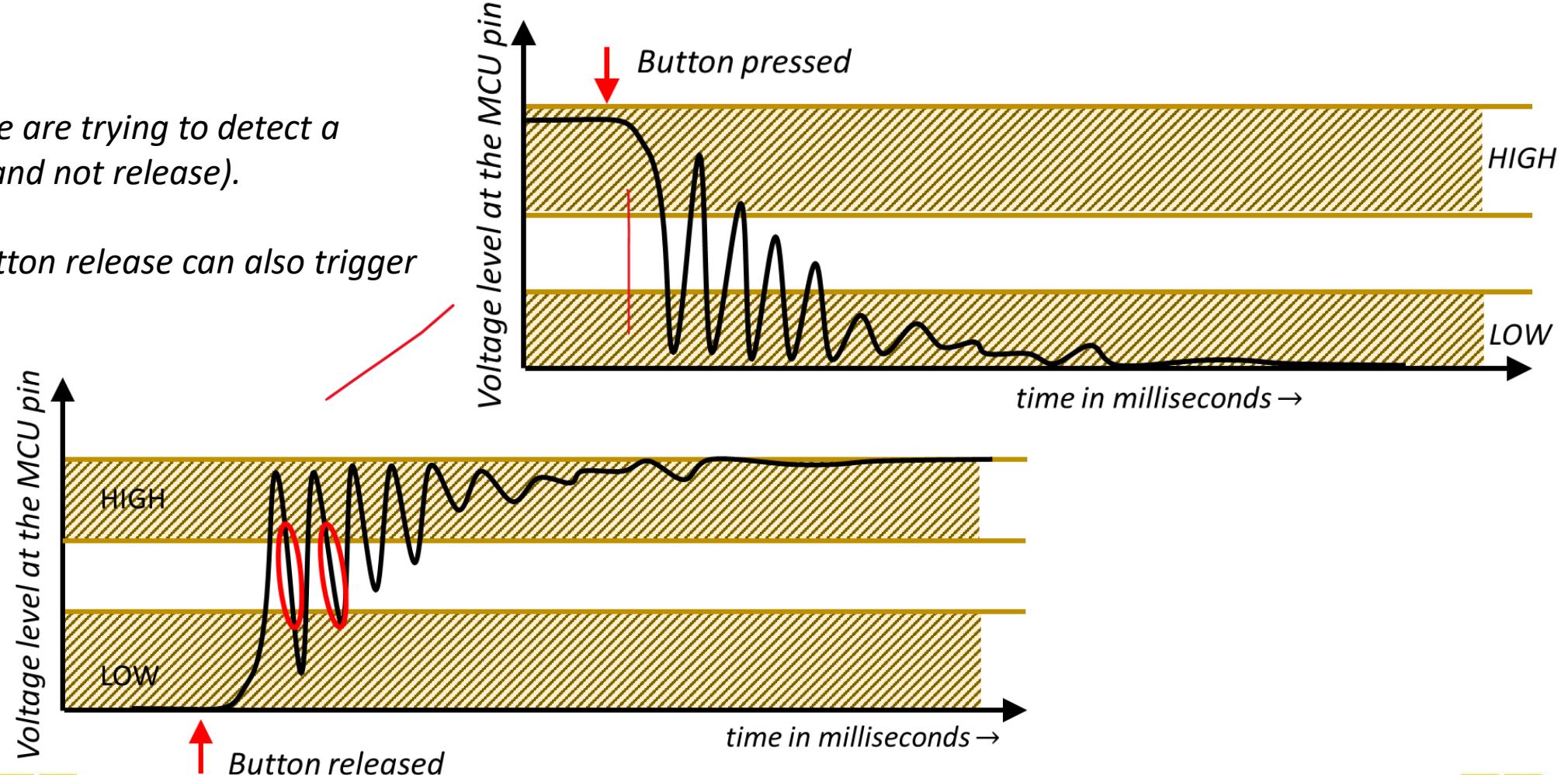
Toggle switches have very long bouncing time and need a longer debouncing period.

Switch Bouncing Problem

- Bouncing occurs in the other direction too.

Remember! We are trying to detect a button press (and not release).

However, a button release can also trigger the interrupt.



Thank You!

Questions?

Email: kamali@ucf.edu

UCF HEC 435 (407) 823 - 0764

<https://www.ece.ucf.edu/~kamali/>

HAVEN Research Group

<https://haven.ece.ucf.edu/>



UNIVERSITY OF
CENTRAL FLORIDA