Switch (de)Bouncing

P.Bernardi

What is switch bounce?

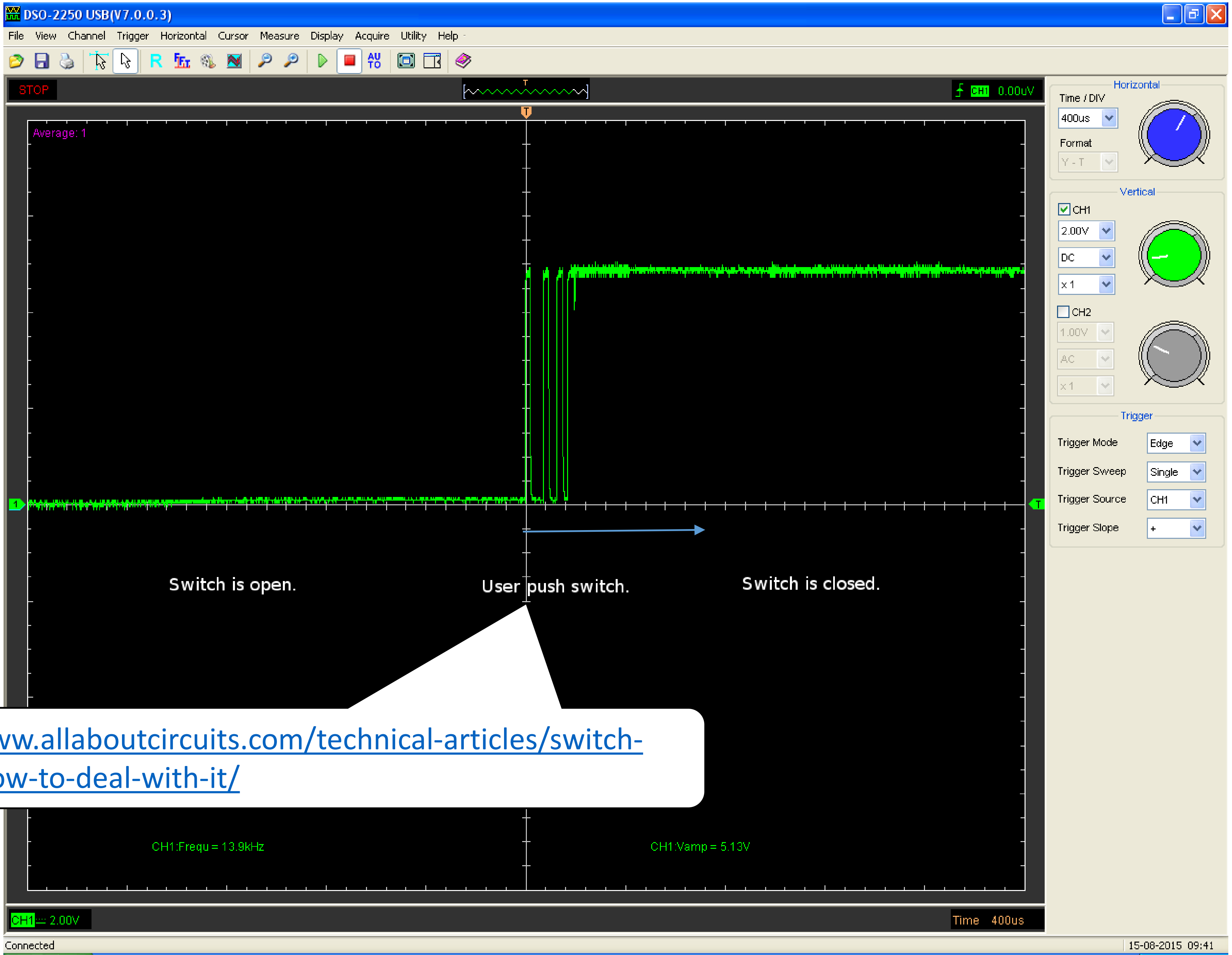
- When you push a button, press a micro switch or flip a toggleswitch, two metal parts come together.
- For the user, it might seem that the contact is made instantly.
- That is not quite correct.

Why a switch bounces

- Inside the switch there are moving parts.
- When you push the switch, it initially makes contact with the other metal part, but just in a brief split of a microsecond.
- Then it makes contact a little longer, and then again a little longer.
- In the end the switch is fully closed.
- The switch is bouncing between in-contact, and not in-contact.

Bouncing measurement

- "When the switch is closed, the two contacts actually separate and reconnect, typically 10 to 100 times over a period of about 1ms." ["The Art of electronics", Horowitz & Hill, Second edition, pg 506.]
- Usually, the SoC works faster than the bouncing, which results in that the hardware thinks you are pressing the switch several times.



Horizontal
Time / DIV: 400us
Format: Y - T

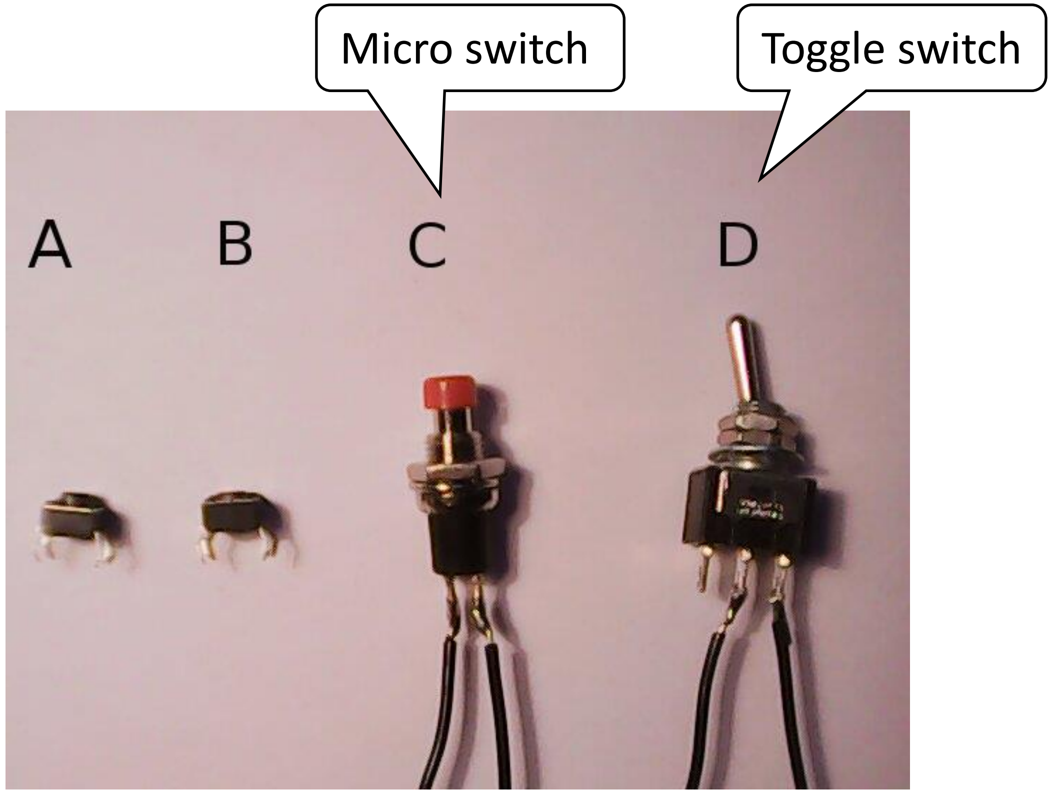
Vertical
CH1 [✓]: 2.00V, DC, x 1
CH2: 1.00V, AC, x 1

Trigger
Trigger Mode: Edge
Trigger Sweep: Single
Trigger Source: CH1
Trigger Slope: +

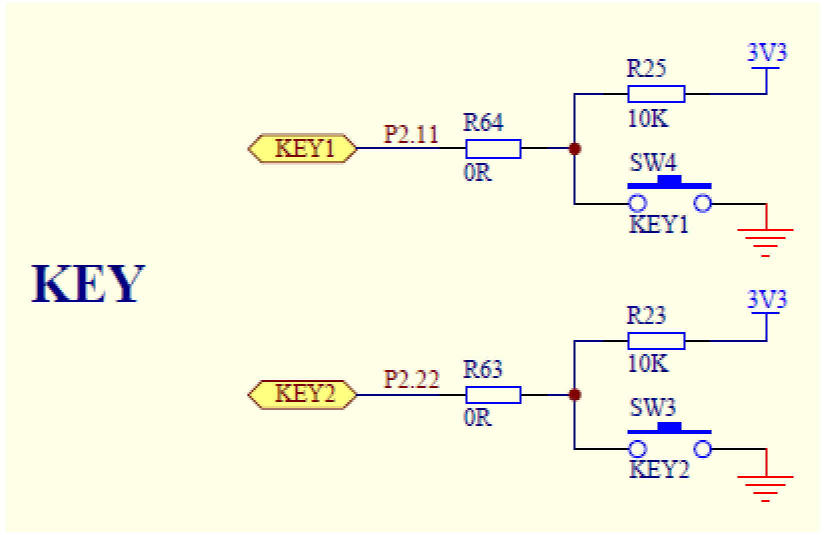
https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

- A nice experiment about 4 switch components is shown at the above URL



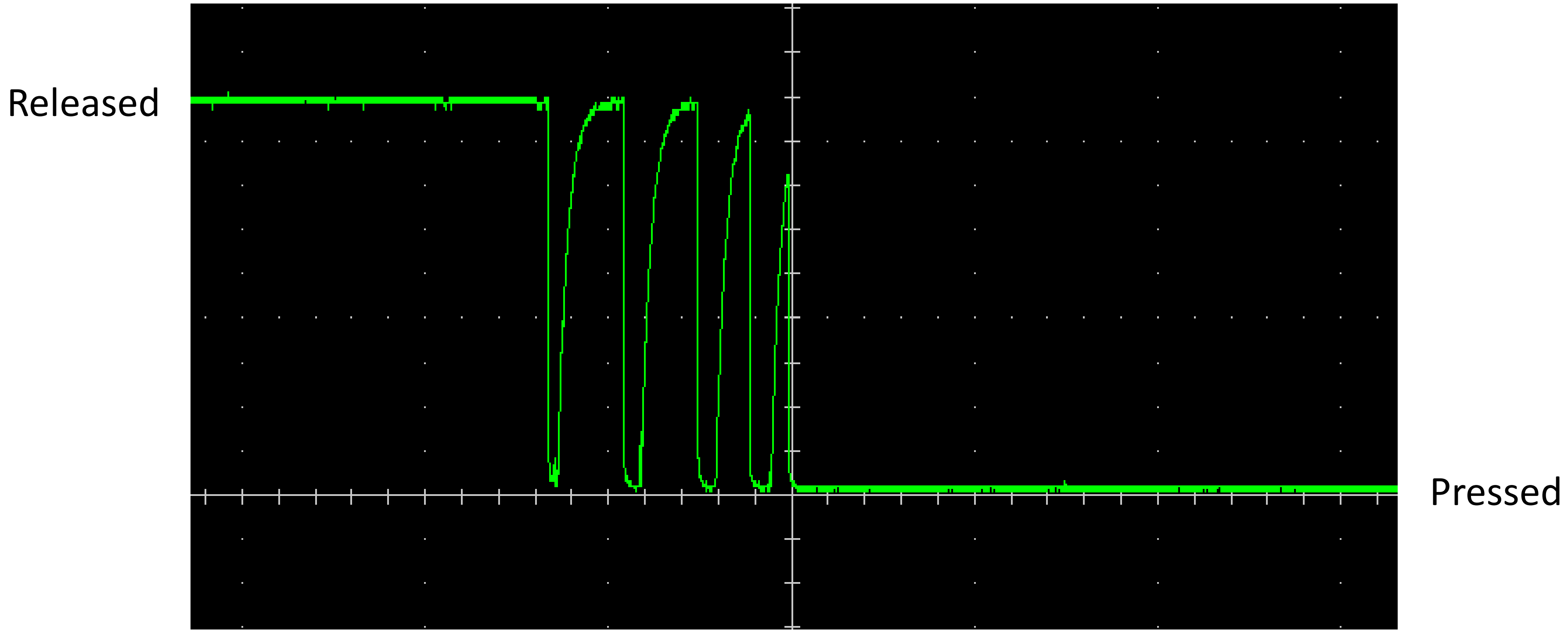
Key behavior



When pressed, the button drives the logic value 0

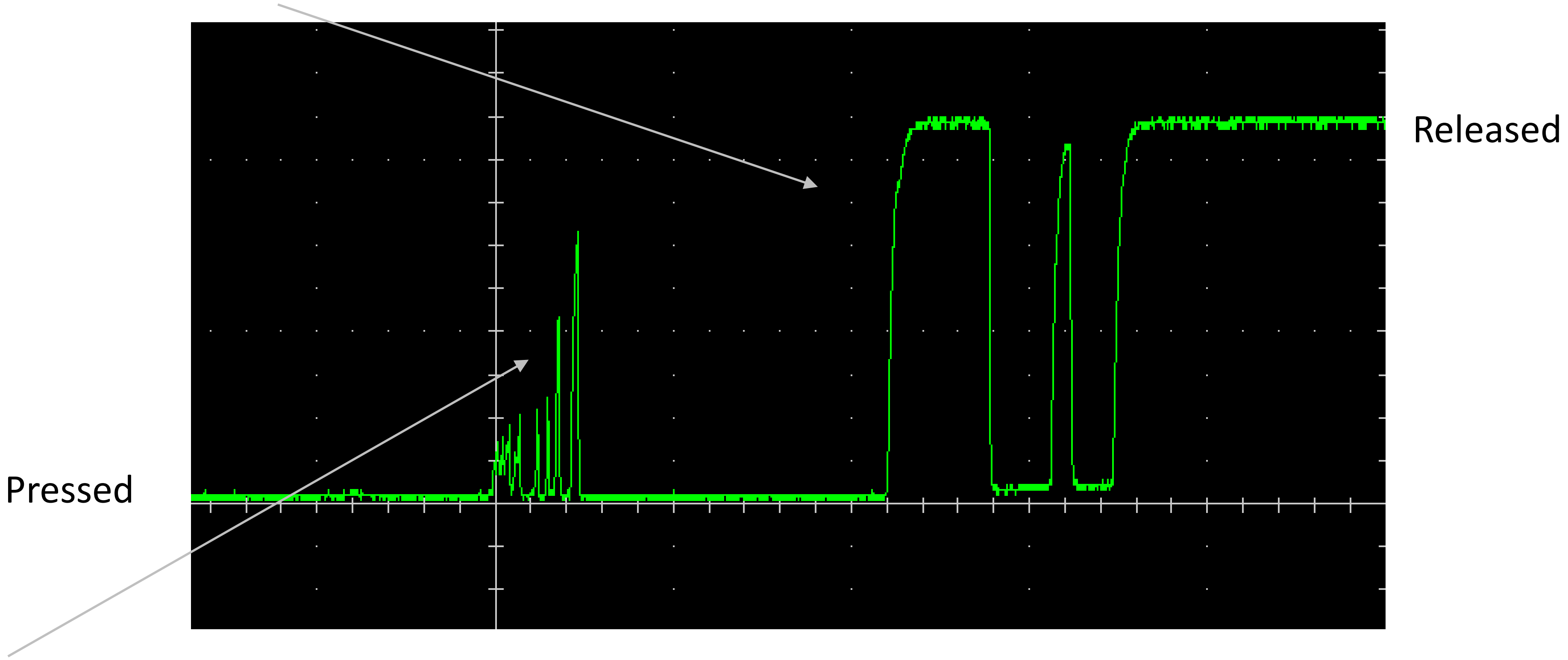
When released, the button drives the logic value 1

Button pulse-low pressure (our case)



Button bouncing example

- Also the release of the button may produce bouncing



- Spikes may also show up (and be triggered during pressure of the button).

A first conclusions

- When working with microcontrollers, we can deal with switch bounce in a different way that will save both hardware space and money.

SW implementation

- A common way to deal with switch bouncing is to re-read the value of the pin after 50ms delay from the first bounce.

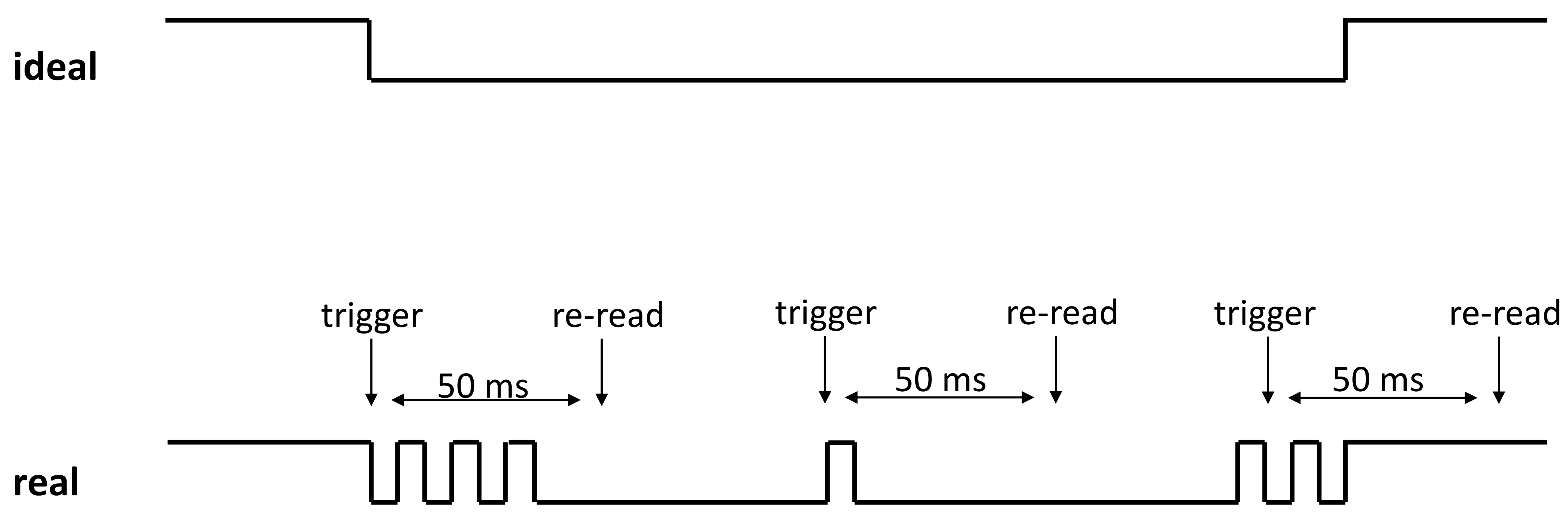
Assumptions

- We prefer using interrupt mode for buttons (this is power efficient as we can enter power down mode since the button is pressed)
- If interrupt is not available, “polling” the value of button related pins is the solution
  - A timer can be used to wake up the system at regular time
- Blocking delay implementations are not desirable
  - SW delay by using “for/while/do-while” empty constructs is considered a very “dirty” technique and deprecated.

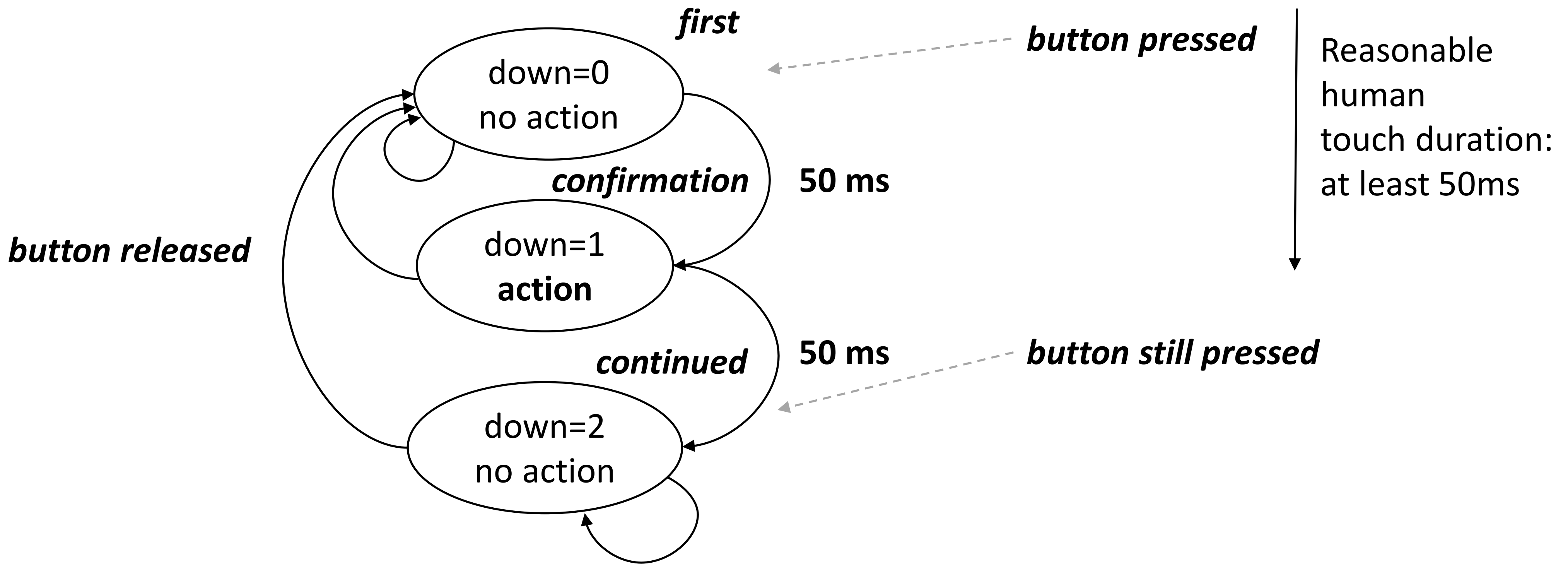
Complications

- If interrupt pin mode selection is selected, the pin may not be directly readable.

Visual solution



Button de-bounce - finite state machine



Repetitive Interrupt Timer

- You may take advantage of the sample project

sample_BUTTON_LED_NVIC_PCON_TIMER_RIT

- that includes the Repetitive Interrupt Timer library.

Exercise

- Experiment switch bouncing with your board and try to mitigate Key bouncing: they must use the external interrupt functionalities

Advanced -> Joystick: implement a «timer controlled polling strategy» also able to mitigate debouncing

Quite Advanced -> can you manage the pressure of many buttons or the contemporary use of buttons and Joystick?

Super-Advanced -> implement button and joystick debouncing by using the RIT only.