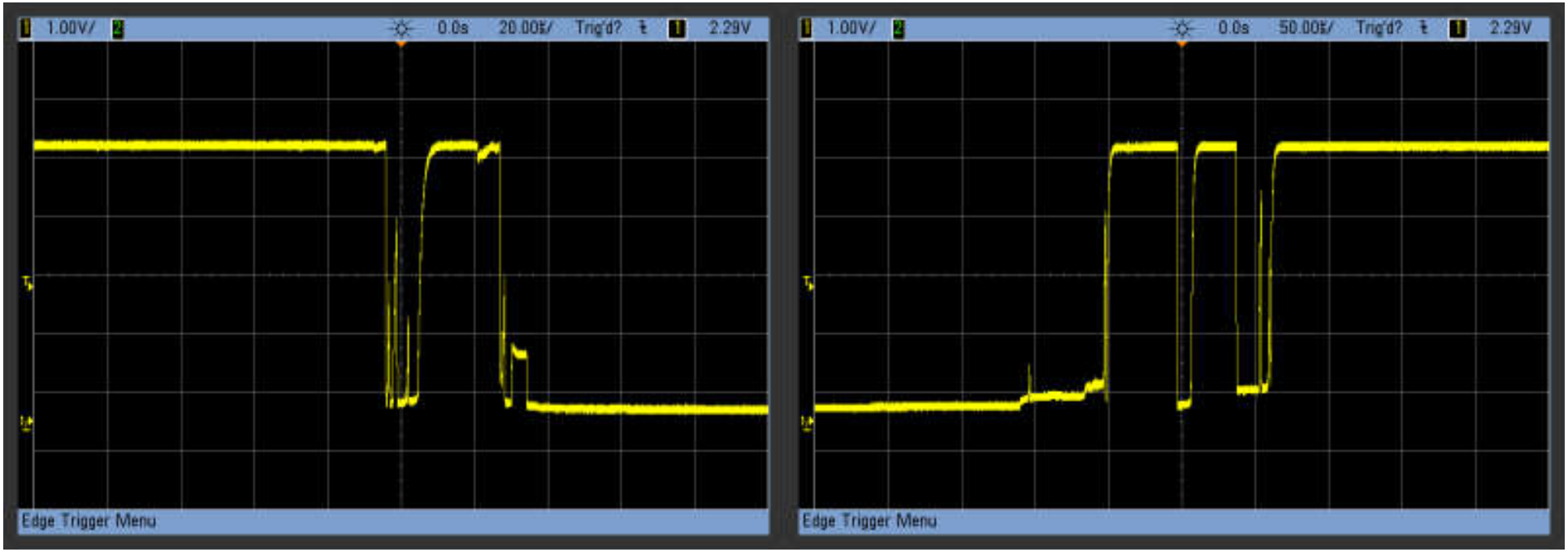
Exercise 7 Key Bouncing

- If you press a key (or when switching any mechanical switch) this typically involves some level of mechanical bouncing
- This is how it looks when measured with an oscilloscope:



Source: https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/

- Bounce times are in the range of ms
  (faster than what we humans can deal with)
- Mercury switches have low bounce times (but toxic)

Source: Wikipedia

APB Series
Backlit Pushbutton Switch

Features/Benefits

- *IP67 sealed*
- *1,000,000 life cycle*
- *Illumination*
- *Threaded or snap-in mounting*
- *RoHS compliant*

Typical Applications

- *Harsh environments*
- *Off-Road*
- *Industrial*
- *Medical*
- *Transportation*
- *Joystick control modules*
- *Gaming*
- *Military*

Specifications

FUNCTION: SPST Momentary
CONTACT ARRANGEMENT: N.O.
MOUNTING TYPE:
Snap-in (no panel seal)
Threaded body (hex nut, lock washer, and panel seal gasket provided)
Torque spec for threaded body: Do not exceed 8-9 in-lbs (0.9-1.0 N.m)

Mechanical

OPERATING LIFE: 1,000,000 cycles
TOTAL TRAVEL: 2.1 ± 0.2 mm
OPERATING POINT 1.55 ± 0.25 mm
OVER TRAVEL: 0.6 min.
OPERATING FORCE: 4N ± 1N standard configuration; other force
option 2N ±
VIBRATION: 10-5
SHOCK: 60g 11r

Electrical

CONTACT RATING:
200mA @ 24 V DC resistive (500,000 cycles)
100mA @ 50 V DC resistive (500,000 cycles)
400mA @ 32 V AC resistive (500,000 cycles)
125mA @ 125 V AC resistive (1,000,000 cycles)
DIELECTRIC STRENGTH: 1000 V AC min.
INSULATION RESISTANCE: 1 G Ω @ 500 VDC
INITIAL CONTACT RESISTANCE 50 mΩ max (without wire leads)
BOUNCE TIME: <5 ms

Materials

HOUSING: PBT
BASE: PBT
CAP: Painted polycarbonate
ACTUATOR:
Illuminated: Clear polycarbonate
INTERNAL SEAL: Silicone rubber
TERMINALS: Copper alloy, gold over silver plating
MOVABLE CONTACT: Copper alloy, gold over silver plating
TERMINAL SEAL: Epoxy
WIRE LEADS: UL1569 Black 22 AWG

Operating Environment

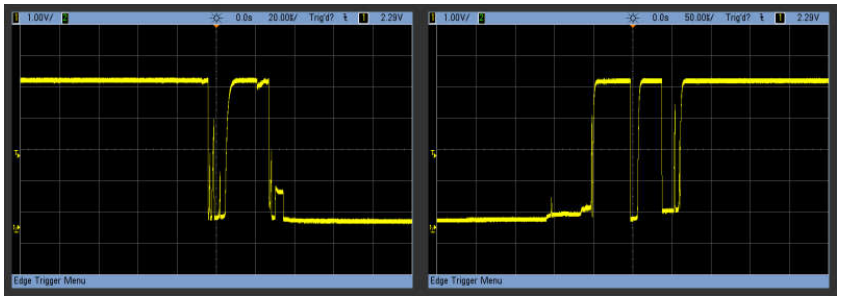
SEALING: IP67 for threaded body
IP65 for snap-in version (no panel seal)

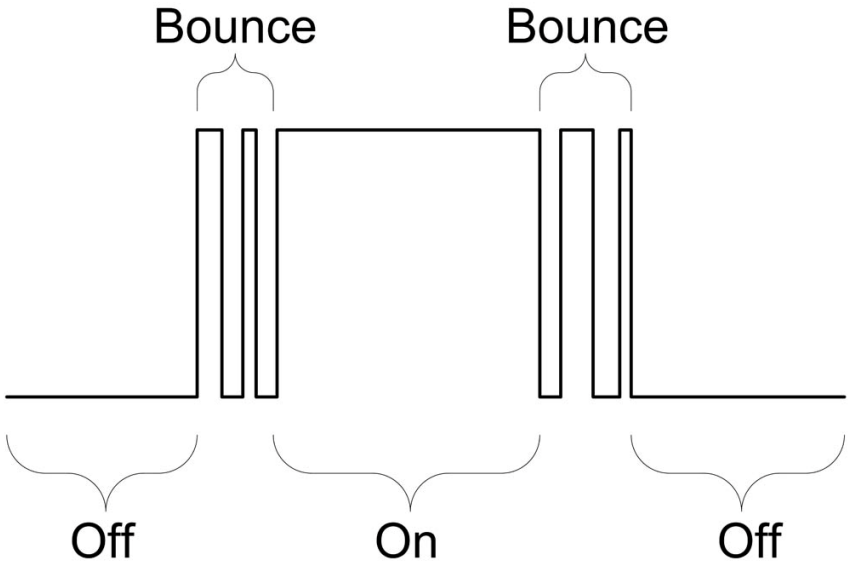
NOTE: Specifications and materials listed above are for switches with standard options. For information on specific and custom switches, please contact Customer Service.

BOUNCE TIME: <5 ms

Exercise 7 Key Bouncing

- In a microcontroller, you will only see digital key states:

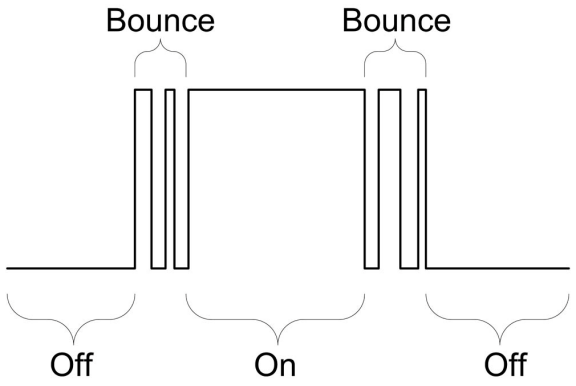




Source: https://stackoverflow.com/questions/20574040/pushbutton-debounce

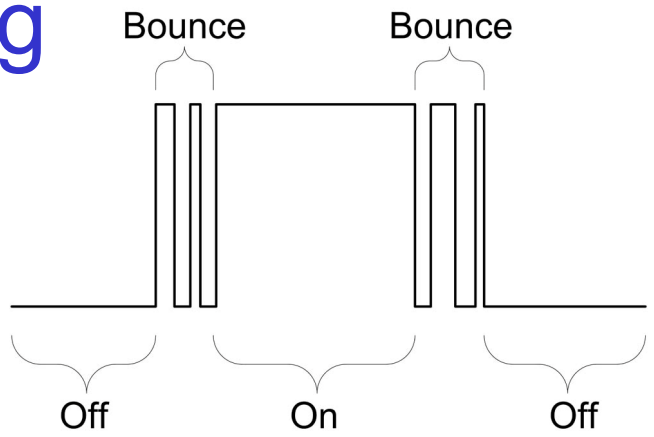
- You have bouncing when pressing and when releasing a button (or switch)

Exercise 7 Key Debouncing



- You have to debounce push buttons (and switches)
- There exist even dedicated debounce chips:
  (e.g. MAX16054) → they cost extra money, so you don't want to use them

Exercise 7 Key Debouncing



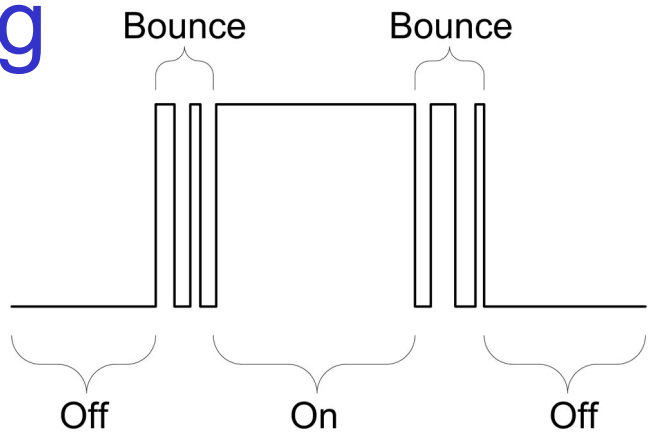
Here debouncing means:

masking key bouncing effects in software

Different strategies

- Infrequent sampling
  (quick and dirty apprach that we don‘t do in COMP22712!)
  - Sampling slower than the bounce time (let‘s say, once every 100ms)
  - Problem: non-deterministic response time
    (the sampling may couse reaction instantanously or delayed)
- Shiftrergister sampling
  - You sample fast (let‘s say, once every 1 ms) and record the last n (e.g. n=8) samples in a shift register
    → you keep a sample history for each button
  - When your sample registers shows continous 1-values (b“1111_1111“) you change the button status to 1, if you sample continous 0-values (b“0000_0000) you change the button status to 0
  - Works quite well but adds some latency to change button states

Exercise 7 Key Debouncing



Here debouncing means:

masking key bouncing effects in software

Different strategies

- Pause sampling after key state change
  - You change key status right away, but you prevent rescaning for the bounce time
  - Requires an individual counter for each button/key
  - Geeky solution: bitslicing
    - Rather than having individual memory addresses for each key, you spread the count value out over multiple registers (memory locations) that are packed with multiple counters
      R0: 1000 0100 0010 0001 (input (added to count values in R1...R3)
      R1: 1000 0100 0010 0000 (LSBs, bit-0, 2^0)
      R2: 1010 0000 0110 0100 (bit-1, 2^1)
      R3: 0000 0100 0000 0000 (MSBs, bit-2, 2^2)
      The blue value (bit-5) is 3 and we add a 1 so after adding we have R3[5]=1, R2[5]=0, R1[5]=0 using Boolean operations