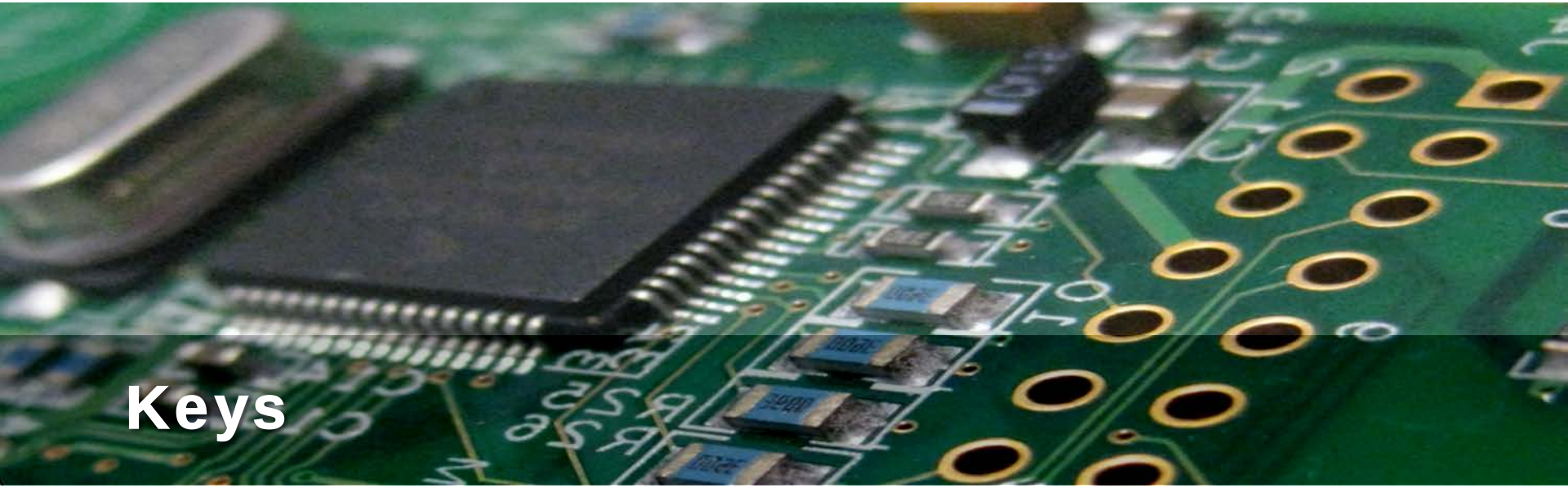
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



Keys

“We are going to use the available switches on the boards as additional way the user can provide input.”

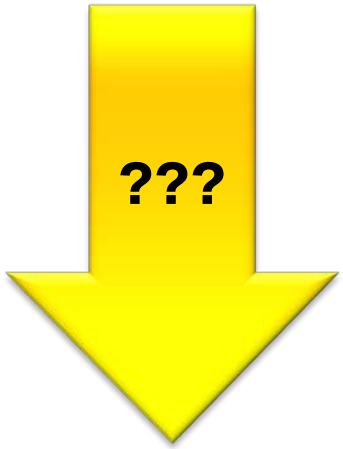
Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Scriptum:
Bouncing Switch

Learning Topics

- Problem: use keys/switches for application

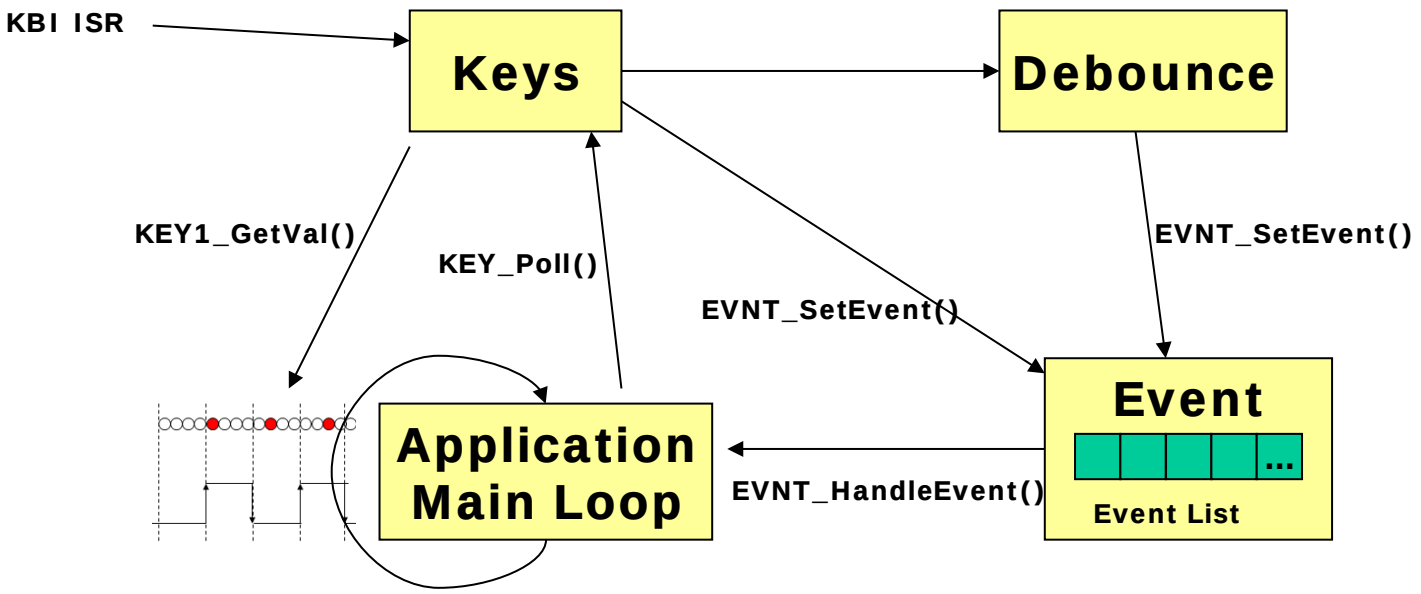
- Keys
  - Hardware/Connectivity
  - Pull-Up, Pull-Down
  - Interrupt
  - Polling
- Software
  - Key driver
  - Control and data flow



Push Buttons

Key Scanning

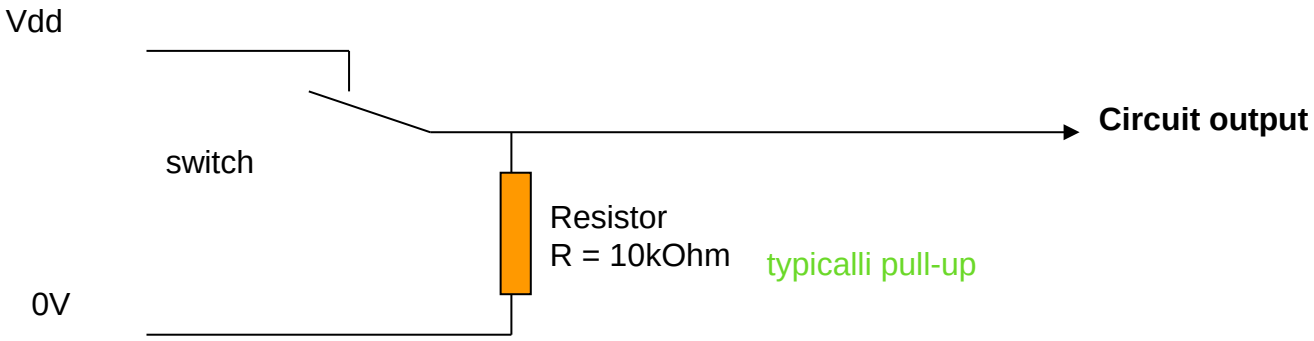
- *Key press detection (Polling and Interrupts)*
- Debounce; long and short key detection



Pull-Up, Pull-Down

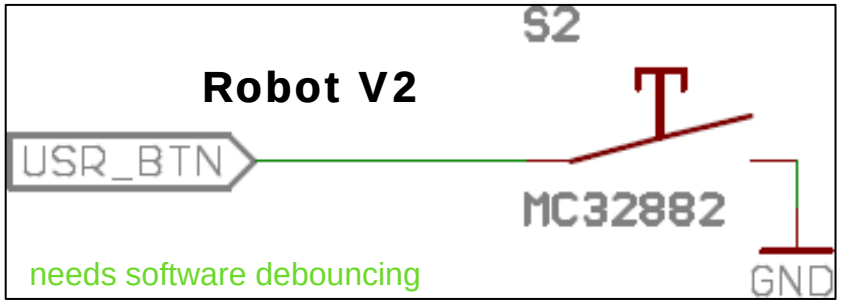
- Open switch?
  - undefined!

- Solution
  - Enforcing of defined logical level
    - Internal circuit (resistor/port configuration)
    - external circuit (resistor)

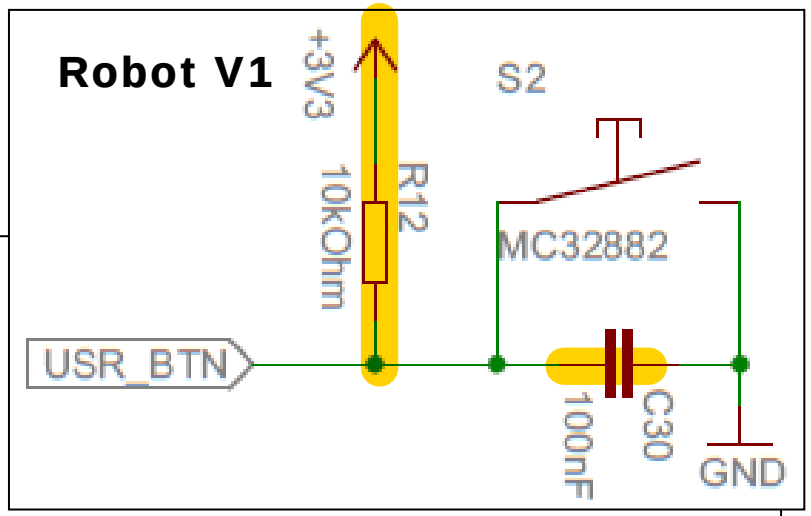


V2 Hardware: Push Button

- No hardware pull-up resistor for USR_BTN
- Use microcontroller internal pull-up



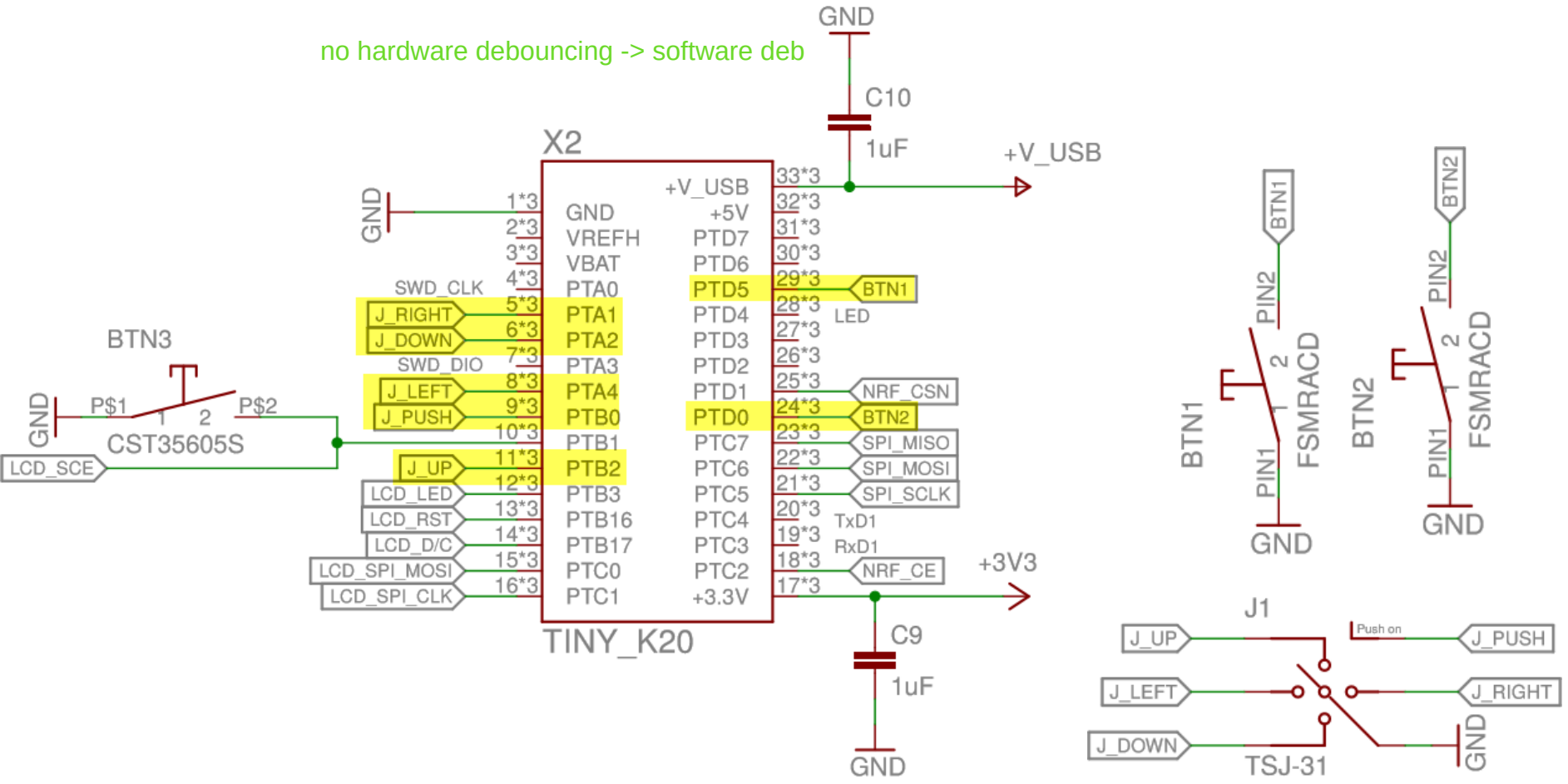
hardware debouncing with a pull-up



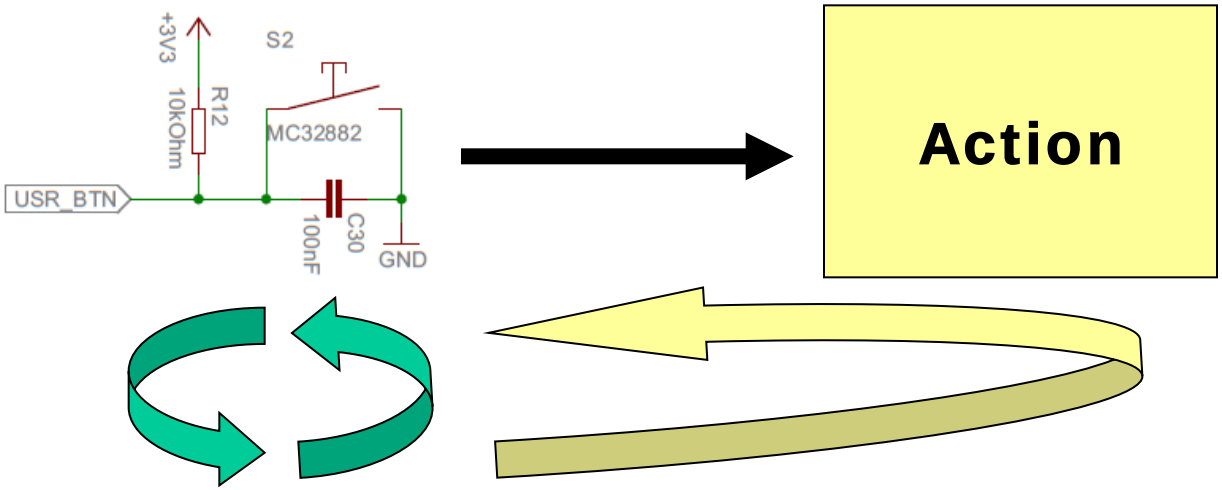

```
#if PL_CONFIG_BOARD_IS_ROBO_V2
  #include "PORT_PDD.h"
#endif

void KEY_Init(void) {
#if PL_CONFIG_BOARD_IS_ROBO_V2
  /* enable and turn on pull-up resistor for PTA14 */
  PORT_PDD_SetPinPullSelect(PORTA_BASE_PTR, 14, PORT_PDD_PULL_UP);
  PORT_PDD_SetPinPullEnable(PORTA_BASE_PTR, 14, PORT_PDD_PULL_ENABLE);
#endif
}
```

Remote Schematics

- No pull-ups and no debouncing capacitors



Keyboard Driver: Solution 1



main loop: 10ms - 50ms

- Realtime Synchronization
- Gadfly Synchronization

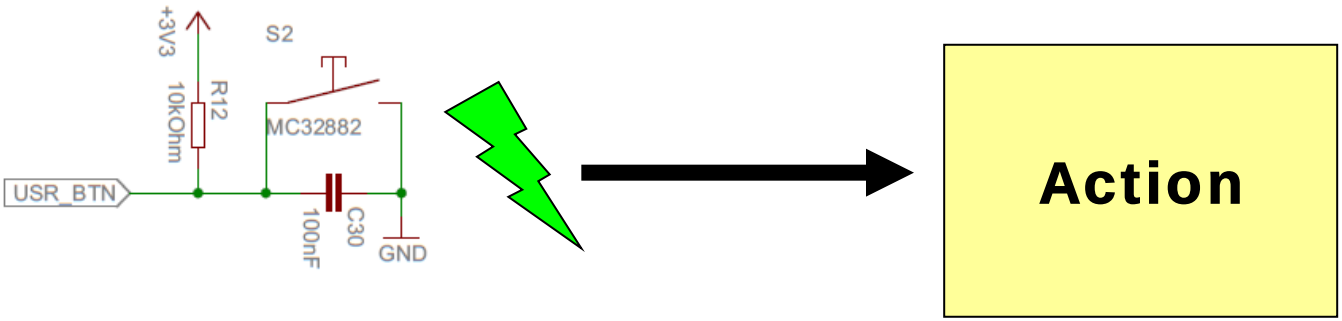
  polling a bit (status bit)

Simple Realtime/Gadfly Synchronization

```
if (SW1_Get()) {
  WAIT1_Waitms(50); /* simple debounce */
  if (SW1_Get()) { /* still pressed? */
    cnt = 0;
    while(SW1_Get()) {
      WAIT1_Waitms(1);
      cnt++; /* measure time */
    } /* wait until released */
    if (cnt<=1000) { /* short press*/
      EVNT_SetEvent(EVNT_SW1_SHORT_PRESSED);
    } else { /* long press*/
      EVNT_SetEvent(EVNT_SW1_LONG_PRESSED);
    }
  }
}
```

blinking an led in this event

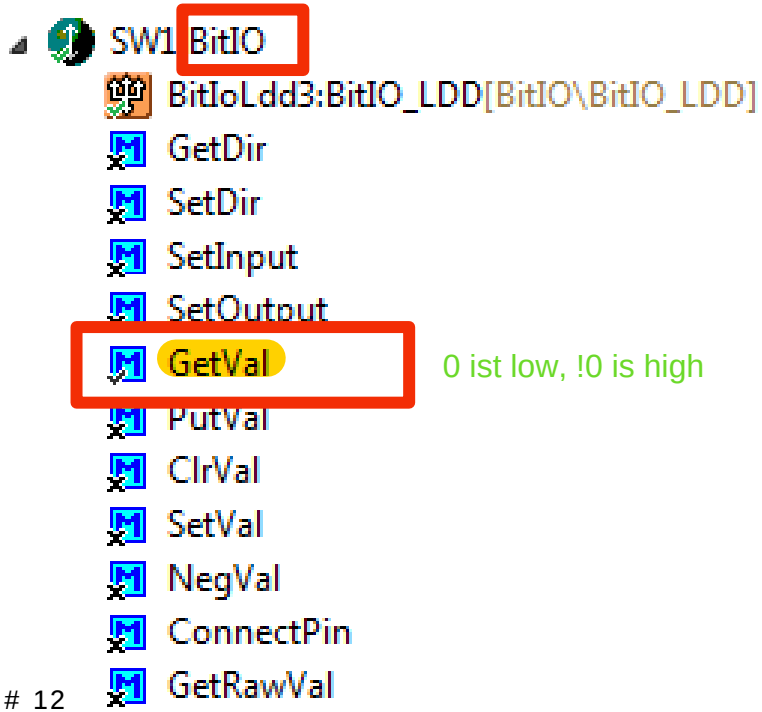
Keyboard Driver: Solution 2



- Interrupt Synchronization
- No Busy Waiting/Polling
- Requires that pin is able to generate interrupt

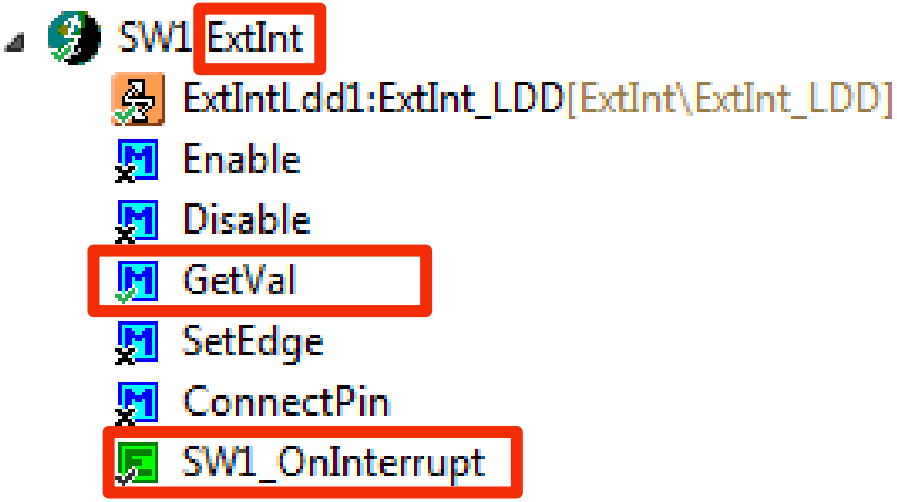
Key Pin Synchronization

- BitIO Component
  - Polling/Gadfly
- ExtInt Component
  - Polling/Gadfly, Interrupt
  - Not every pin has interrupt capability!

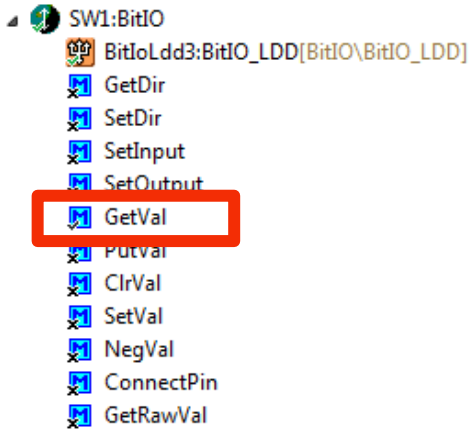


0 ist low, !0 is high

falling, rising edge



Control and Data Flow with Polling



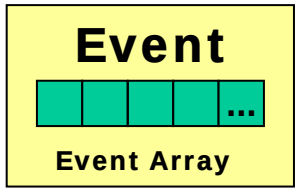

```
callback(EVNT_Handle event):
  switch(event) {
    case EVNT_SW1_PRESSED:
      print("SW1 pressed");
```

Main Application Loop:

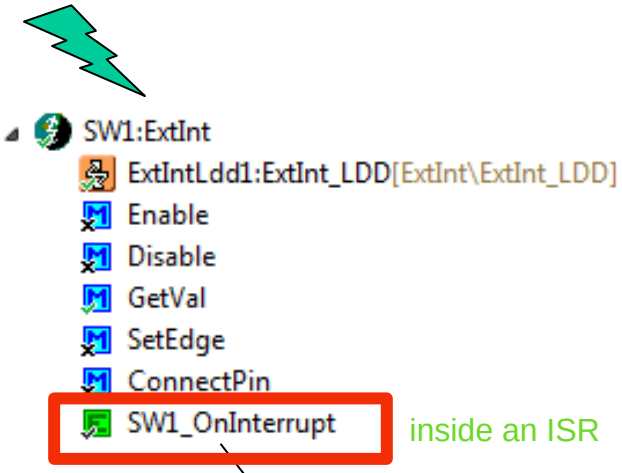
```
for(;;) {
  KEY_Scan();
  EVNT_HandleEvent(callback);
}
```

Keys.c

```
KEY_Scan(void):
  if (SW1_GetVal()) {
    EVNT_SetEvent(EVNT_SW1_PRESSED);
```



Control and Data Flow with Interrupts



inside an ISR

Events.c

```
SW1_OnInterrupt():
  KEY_OnInterrupt(KEY_BTN1);
```

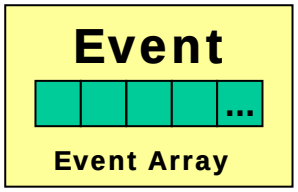
Keys.c

```
KEY_OnInterrupt(KEY_Buttons button):
  if (button==KEY_BTN1) {
    EVNT_SetEvent(EVNT_SW1_PRESSED);
```

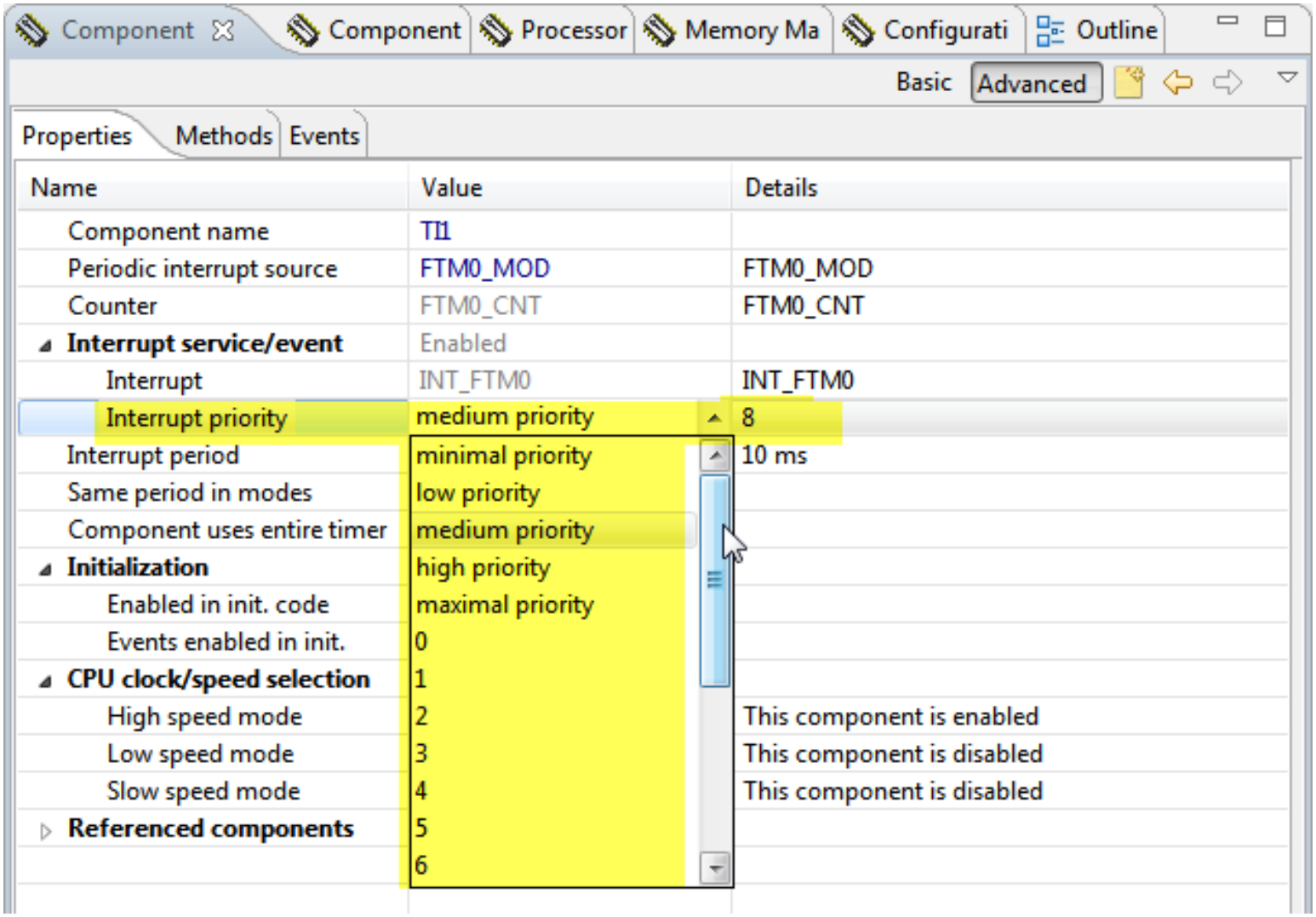
```
callback(EVNT_Handle event):
  switch(event) {
    case EVNT_SW1_PRESSED:
      print("SW1 pressed");
```

Main Application Loop:

```
for(;;) {
  EVNT_HandleEvent(callback);
}
```

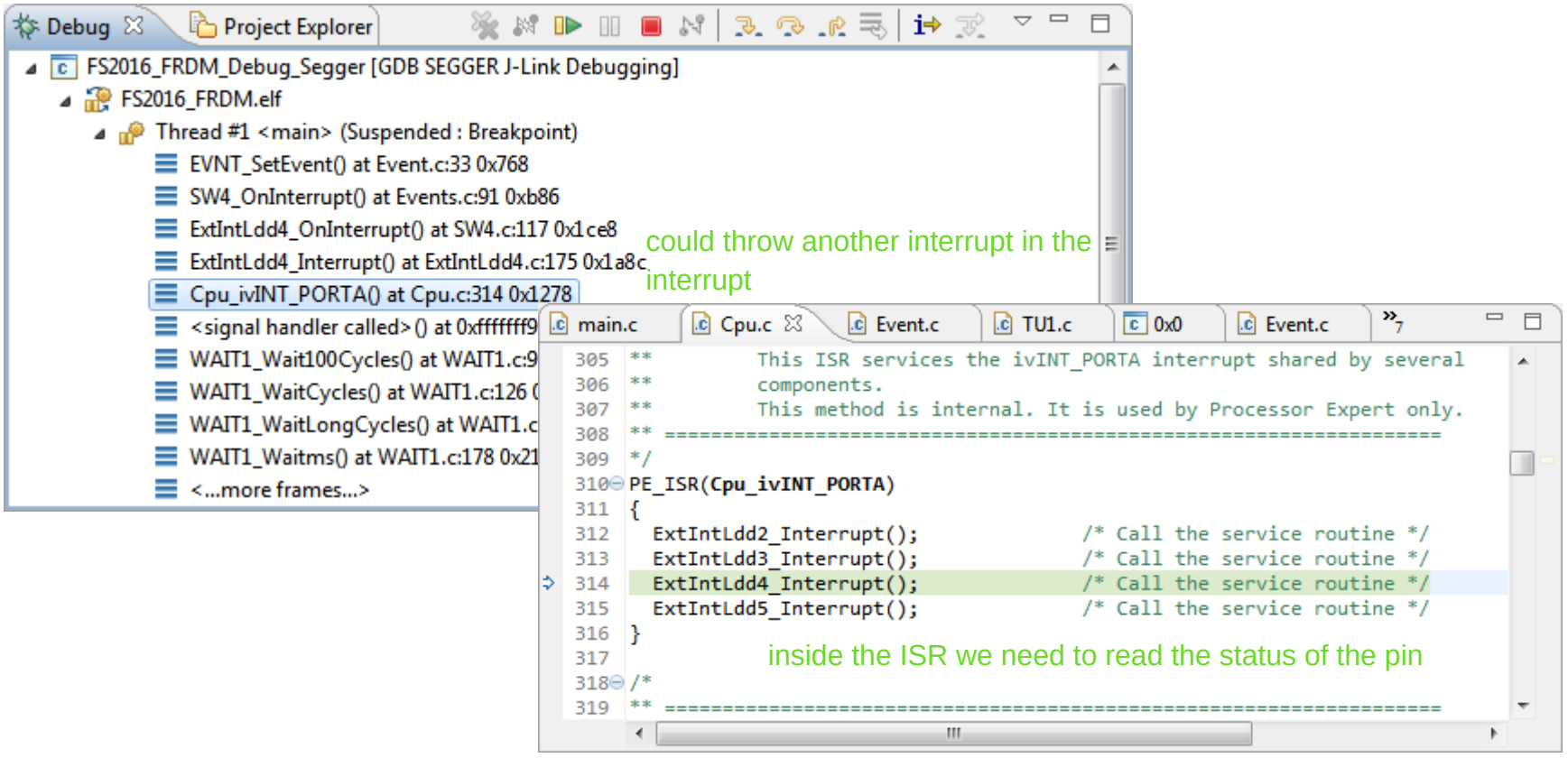


Interrupt Priorities



Port Interrupt Sharing

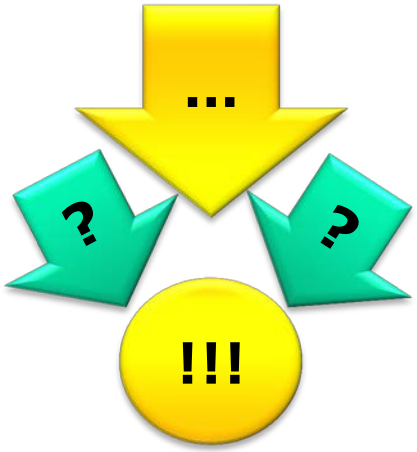
- Cortex-M0+: only 32 interrupt sources
- One Interrupt for all port pins
- Need to poll/check in ISR which pin triggered ISR



Summary

- *Problem: we want to use keys/switches for our application*

- Keys
  - Pull-up/pull-down
- Synchronization
  - Realtime
  - Gadfly
  - Interrupts
- Creating Events for keys



Lab: Keys

- Robot: 1 push button
- Remote: 4+1+2 push buttons
- Keys.h, Keys.c
  - Implement polling for all keys
  - Add interrupts
  - Create events for key presses