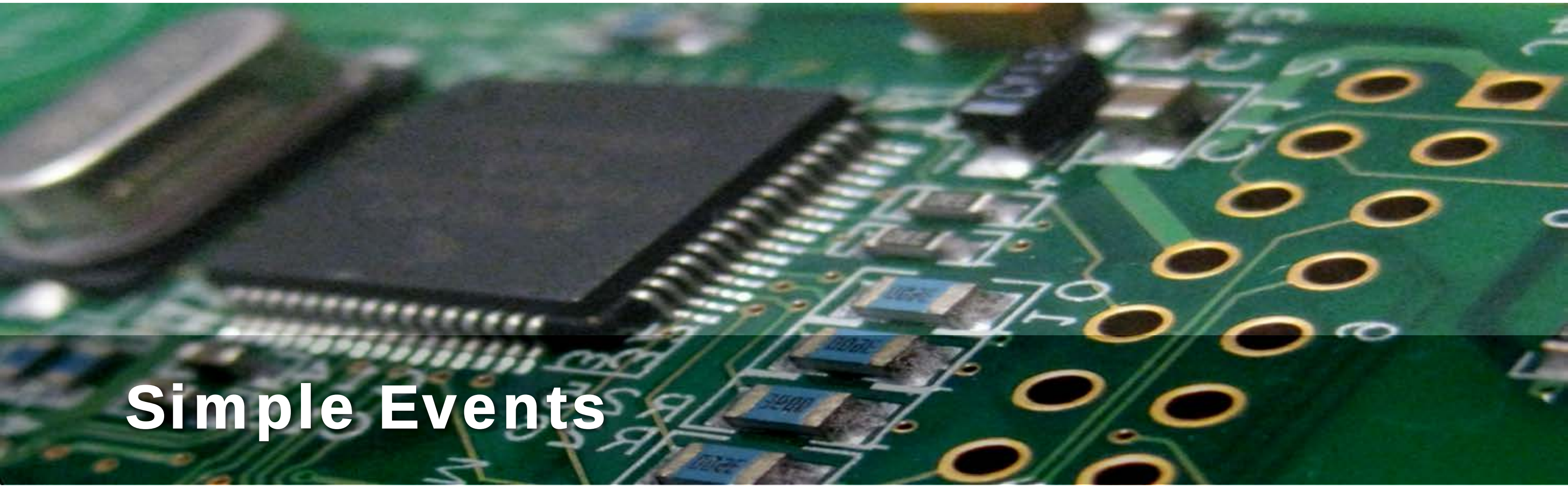


Simple Events

“We need a way to deal with simple events to be processed by the main application.”

problem: now it's not reentrant -> we need to implement that

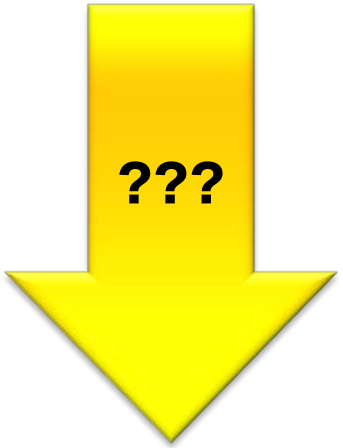
Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Scriptum:
Events

Learning Goals

- Problem: Infrastructure for Synchronization: Event Module

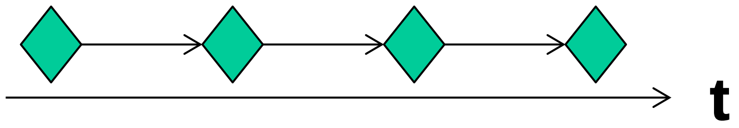
- Polling vs. Interrupts
- Time Synchronization
- Priorities
- Reentrancy

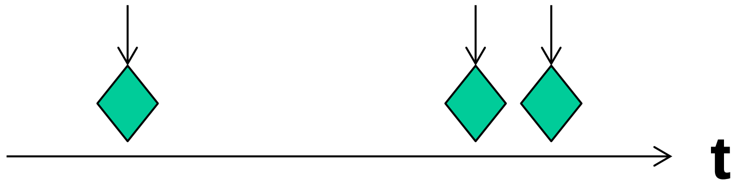


Events

- **Synchronous** Events
  - Timer interrupt
  - Periodic Task output
- **Asynchronous** Events
  - Button pressed
  - Transceiver packet received
  - Beep after button press
- Need Infrastructure
  - Set/clear/check if event happened
- Implementation
  - RTOS
  - 'Flags'
  - Possible implementation
    - Queue, List, **Array**

we only recognize that there is an Event

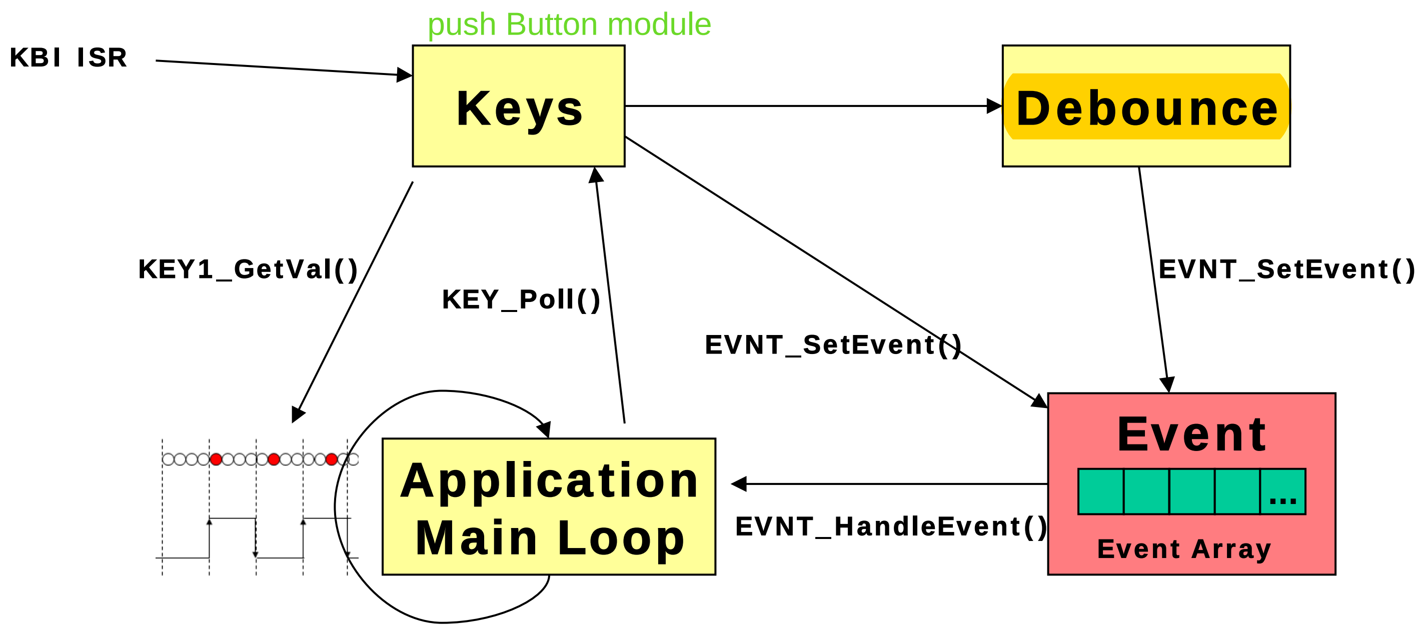




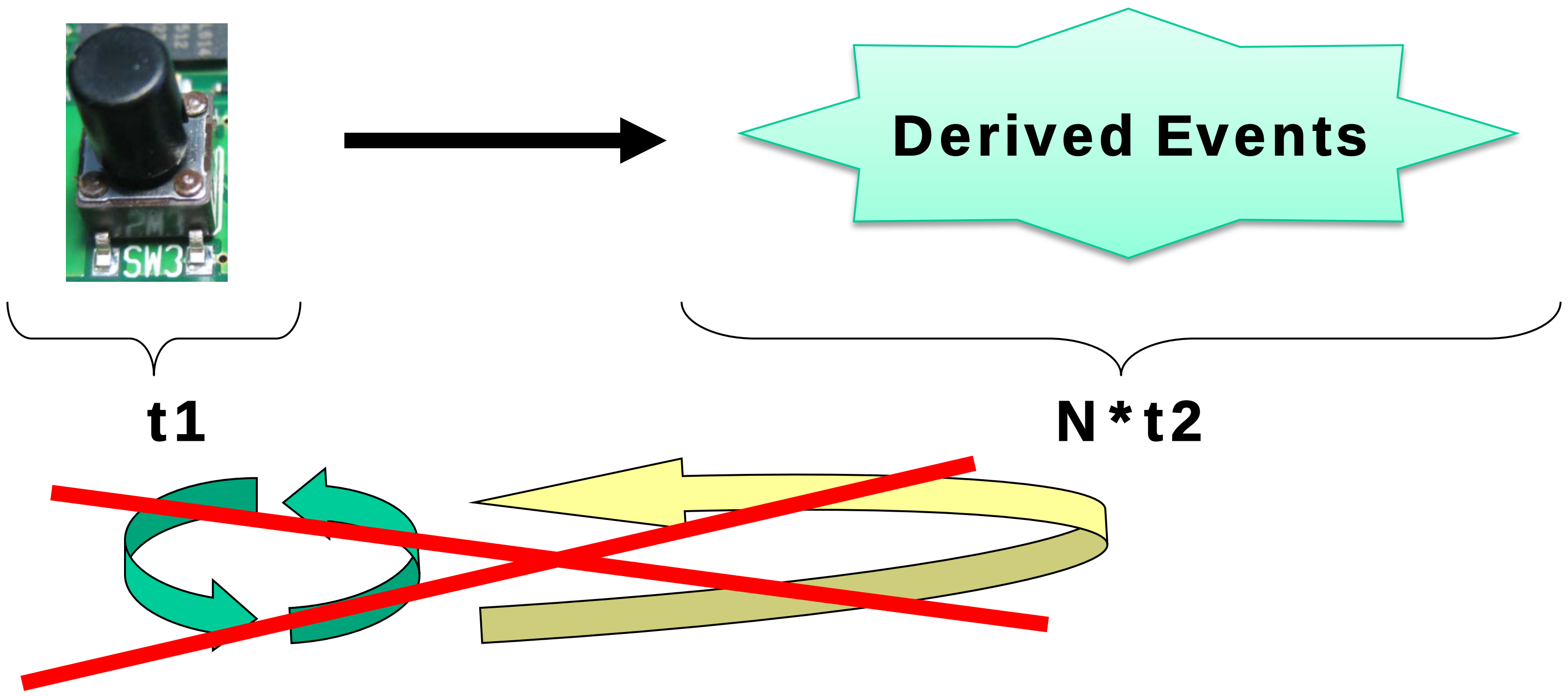
Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN
Technik & Architektur

Lab Task Outlook

- **Event** Module Implementation (Bit Arrays)
- **Trigger** Module Implementation (Interrupts)
- **Keys** Module Implementation (KBI, Function Pointers)
- **Debounce** Module Implementation (FSM)



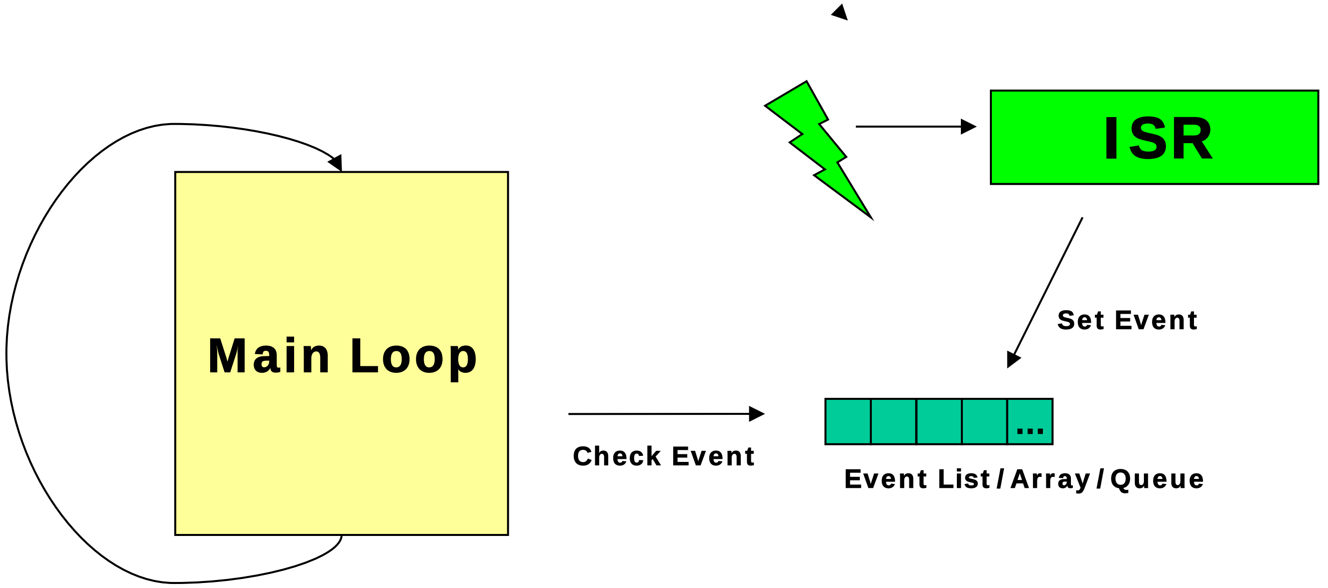
Events and Derived Events



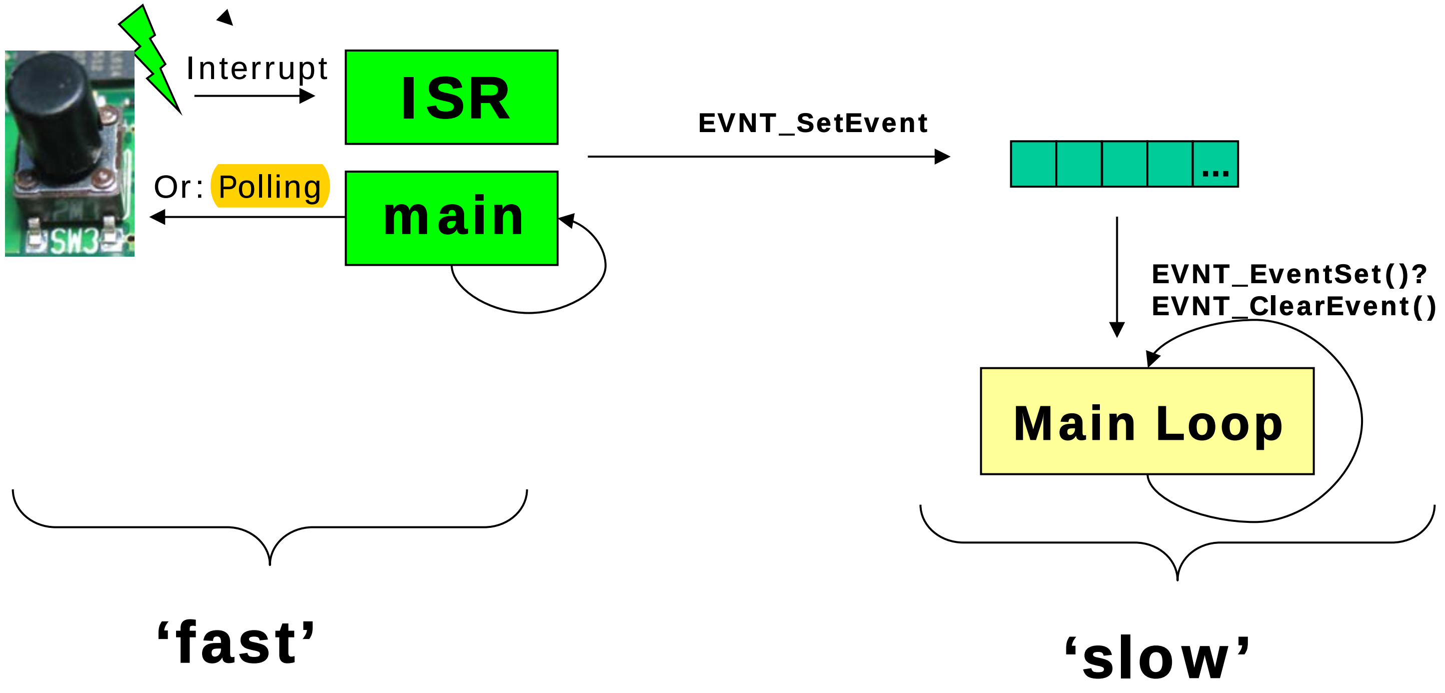
- Execution of time consuming algorithms
- Sequence of actions (button press ➔ start countdown)
- Nesting (button press ➔ beep + blink LED)
- Need to decouple 'source of event' and 'actions'

Interrupt Execution Speed

- ISR: as efficient and straight forward as possible
- Possible approach: Event Loop/Handler
- Main loop does the 'heavy' workload
- Interrupt Service Routines: Create/Set events (flags)



Decoupling Event and Processing

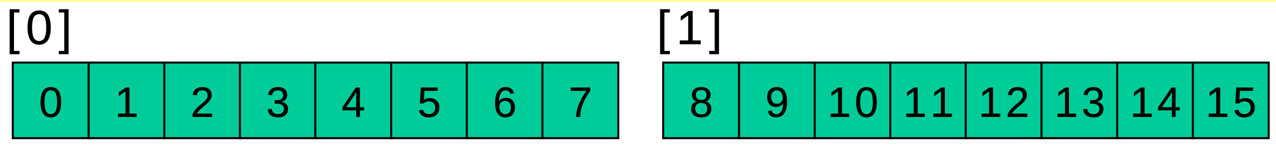


doesn't have to be immediately

EVNT Array

- Array of Bytes:

```
static uint8_t EVNT_Events[((EVNT_NOF_EVENTS-1)/8)+1];
```



- Set event:

```
EVNT_Events[event/8] |= 0x80>>(event%8);
```

- Considerations:
  - Bit order (Little Endian, Big Endian, ...)
  - Size of base memory unit: uint8_t, uint16_t, ...

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

Storing Events?

- Idea
  - Using as few memory as possible
  - Using event 'flags'
  - ➔ mapping from ‘numbers’ to bits/flags

```
typedef uint8_t EVNT_Handle; /*!< We can support up to 256 different events */

#define EVNT_INIT              0
  /*!< System Initialization Event */
#define EVNT_SW1_PRESSED       1
  /*!< SW1 pressed */
#define EVNT_SW2_PRESSED       2
  /*!< SW2 pressed */
#define EVNT_SW3_PRESSED       3
  /*!< SW3 pressed */
#define EVNT_SW4_PRESSED       4
  /*!< SW4 pressed */
#define EVNT_LED_HEARTBEAT     5
  /*!< LED heartbeat */
#define EVNT_NOF_EVENTS        6
  /*!< Must be last one! */
```

Implications if using 'enums'?

Numbering can be priority!

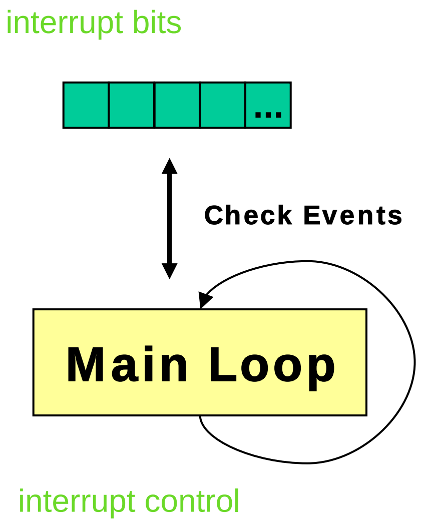
Event Enumeration Type

- Symbolic names instead of #define
- Sentinel at the end
    - Gives Number of Event items
    - Used for bit array size definition

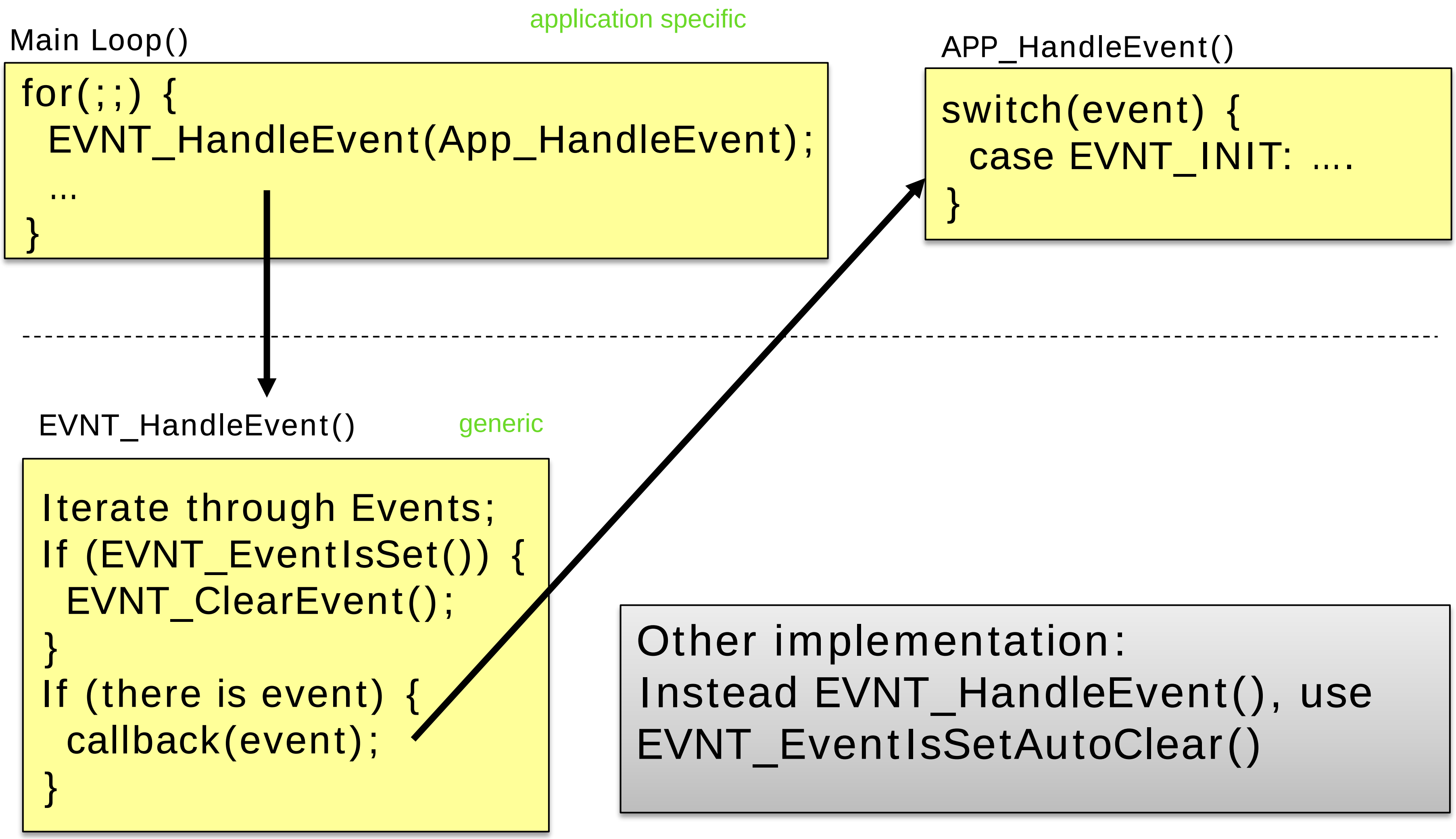
```
typedef enum {
  EVNT_INIT,              /*!< System Initialization Event */
  EVNT_SW1_PRESSED,   /*!< SW1 pressed */
  EVNT_SW2_PRESSED,   /*!< SW2 pressed */
  EVNT_SW3_PRESSED,   /*!< SW3 pressed */
  …
  EVNT_NOF_EVENTS      /*!< Must be last one! */
} EVNT_Handle;
```

Handling Events from Main Loop

- Extract Event (e.g. Loop)
  - See if there is an event
  - Event 'number' or bit position could be used as priority
  - Extract bit/event
- Handle Event (e.g. Switch)
  - Act according to event (e.g. flash LED's)
- Advantage: simple
- But:
  - Long if/else/switch
  - Order of event handling needs to be defined
  - Need to protect against concurrent access!



EVNT_HandleEvent()



```
for(;;) {
  EVNT_HandleEvent(App_HandleEvent);
  ...
}
```

```
switch(event) {
  case EVNT_INIT: ....
}
```

```
Iterate through Events;
If (EVNT_EventIsSet()) {
  EVNT_ClearEvent();
}
If (there is event) {
  callback(event);
}
```

Other implementation:
Instead EVNT_HandleEvent(), use EVNT_EventIsSetAutoClear()

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

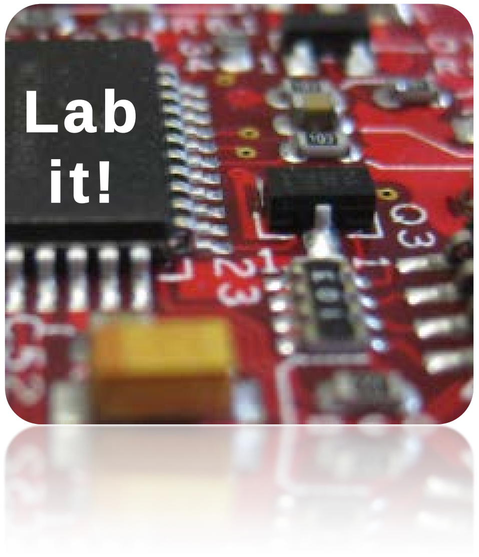
EVNT Interface

```
void EVNT_SetEvent(EVNT_Handle event);

void EVNT_ClearEvent(EVNT_Handle event);

bool EVNT_EventIsSet(EVNT_Handle event);

bool EVNT_EventIsSetAutoClear(EVNT_Handle event);

void EVNT_HandleEvent(
      void (*callback)(EVNT_Handle)
      bool clearEvent);

void EVNT_Init(void);

void EVNT_Deinit(void);
```

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

Lab: Event Module

- Event.c/Event.h
- Platform.c
  - EVNT_Init()/Deinit()
- Application.c
  - APP_HandleEvent
    - EVNT_INIT: perform initialization indication (e.g. flashing LED's)



sizeof() return the numbers of bytes