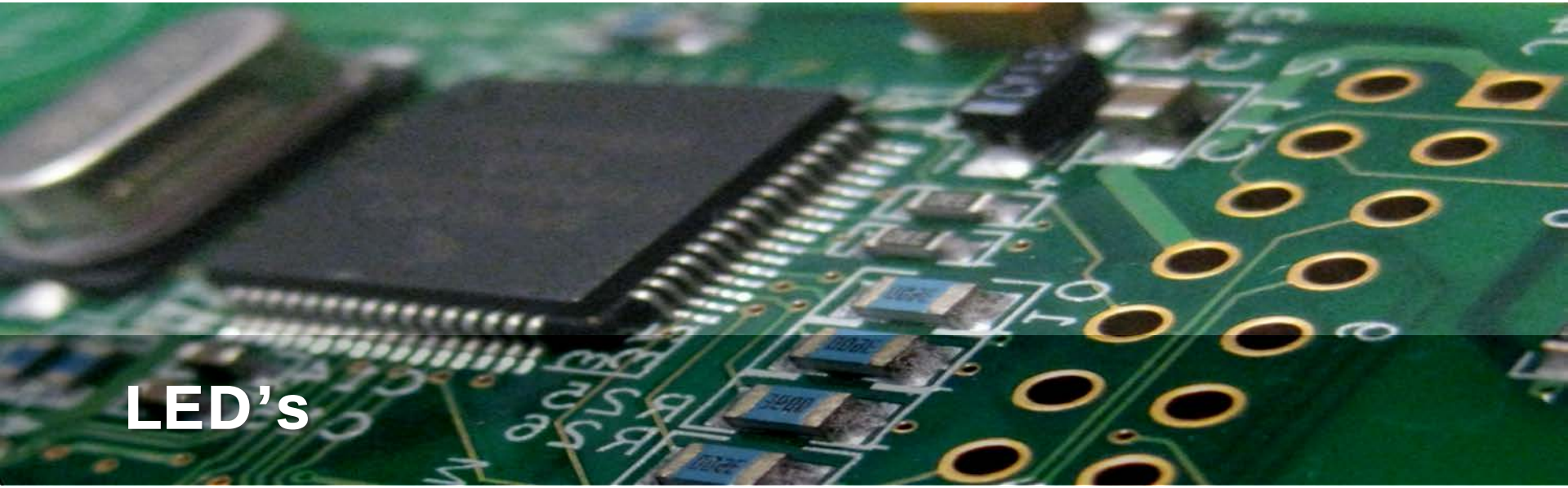
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



LED's

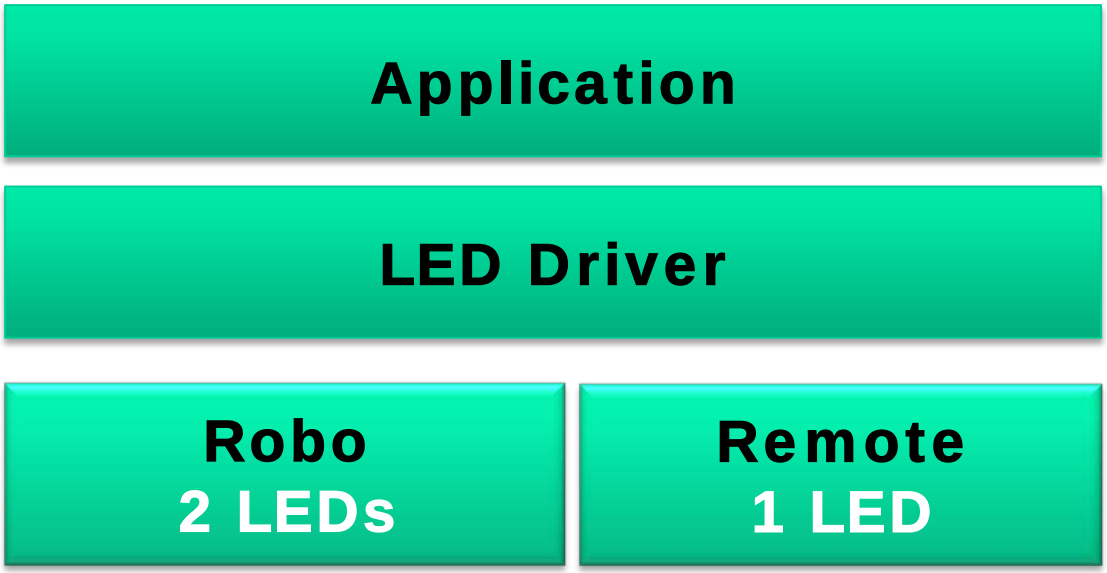
"Now let's make our LED driver perfect. That should be easy."

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

Scriptum:
LED

Implementing a Driver for LEDs

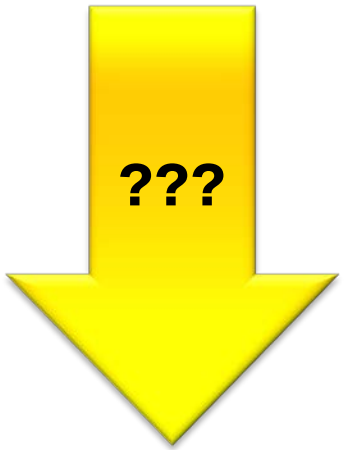
- Use BitIO/LED component as hardware abstraction
- Support for different number of LEDs
- LED Driver should hide implementation details



Learning Goals

- Problem: LED‘s with a microcontroller

- Goal
  - ‘same’ driver for multiple platforms
- LED
  - Characteristics
  - Resistor
  - Wiring
- Microcontroller
  - Block diagram
  - Ports
  - Efficiency
  - Portability



INTRO LED Driver

- LED's, together or individually controlled
  - On/Off
  - Neg
  - Get/Put
  - Init/Deinit
  - Open/Close
- We need
  - Pins
  - Pins as output
  - Application
    - Configuration/Setup
    - Control
    - Status

LED's, Implementation 1

- Goal
  - One Function
  - Multiple LED's

- Problem
  - Multiple LED's as parameters?

```
void LED_On(bool LED0, bool LED1, bool LED2, bool LED3);

void main(void) {
  LED_On(TRUE, FALSE, TRUE, TRUE);
}
```

LED's, Implementation #2

- One argument
- LED's are encoded
  - LED0: 0x01
  - LED1: 0x02
  - LED3: 0x04
  - LED4: 0x08

```
void LED_On(uint8_t LEDs);

void main(void) {
  LED_On(0x01+0x04);
  LED_On(0x02|0x08);
}
```

LED's, Implementation #3

- One argument
- LED's masks definded as symbols

```
#define LED_LED0  0x01
#define LED_LED1  0x02
#define LED_LED2  0x04
#define LED_LED3  0x08

void LED_On(uint_8 LEDs);

void main(void) {
  LED_On(LED_LED0|LED_LED3);
}
```

LED's, Implementation #4

- One argument
- LED's used in a symbolic way
- LED's as type

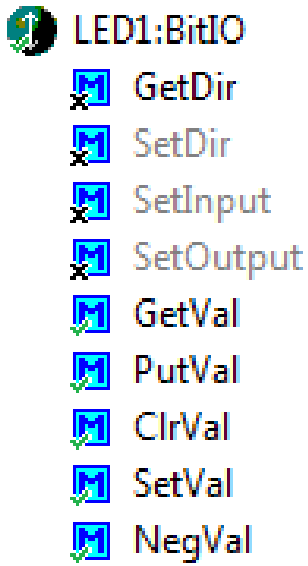
```
typedef enum {
  LED_LED0 = (1<<0),
  LED_LED1 = (1<<1),
  LED_LED2 = (1<<2),
  LED_LED3 = (1<<3)
} LED_Set;

void LED_On(LED_Set LEDs);

void main(void) {
  LED_On(LED_LED0|LED_LED3);
}
```

LED's, Implementation #5

- Interface for each LED
  - Flexible (anode/cathode)
  - For few LED's

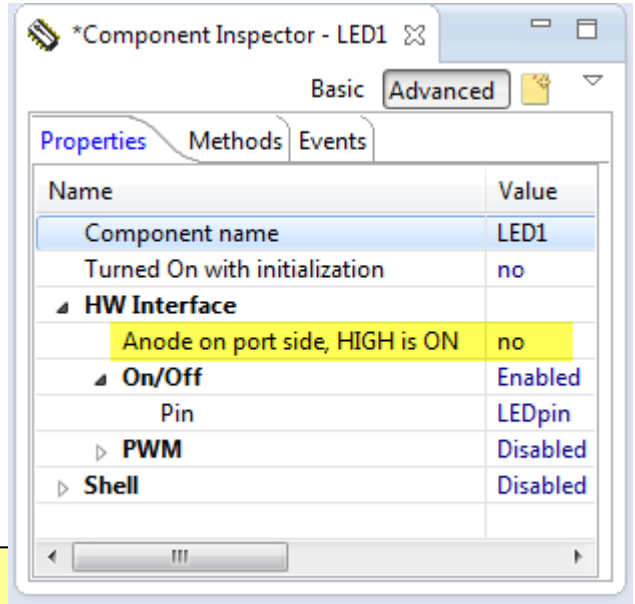
```
void LED1_On(void);
void LED1_Off(void);
void LED1_Neg(void);
bool LED1_Get(void);
void LED1_Put(bool);

void LED2_On(void);
void LED2_Off(void);
void LED2_Neg(void);
bool LED2_Get(void);
void LED2_Put(bool);
```

```
#define LED1_On() LED1_ClrVal()
```



Microcontroller Pin on Anode or Cathode?

```
#if PL_LED_CATHODE_PIN
  #define LED1_On() LED1_ClrVal()
#else
  #define LED1_On() LED1_SetVal()
#endif
…
```




```
#if PL_LED_CATHODE_PIN
  #define LED_TURN_ON(nr)  (LED##nr##_ClrVal())
#else
  #define LED_TURN_ON(nr) (LED##nr##_SetVal())
#endif

#define LED1_On()   (LED_TURN_ON(1))
#define LED2_On()   (LED_TURN_ON(2))
```

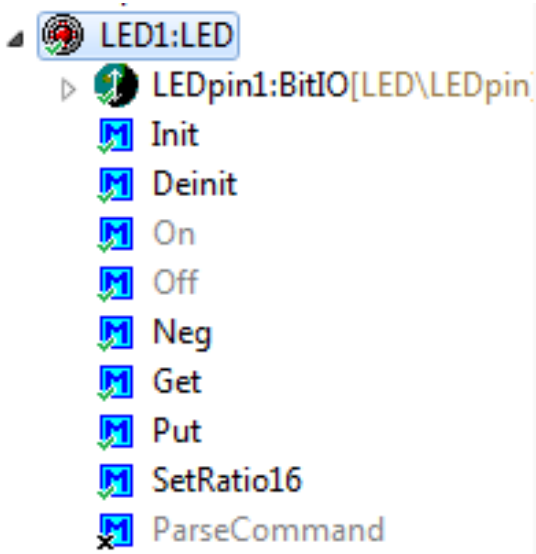
Driver Interface

```
void LED1_On(void);
void LED1_Off(void);
void LED1_Neg(void);
bool LED1_Get(void);
void LED1_Put(bool);


void LED1_Open(void);
void LED1_Close(void);


void LED1_Init(void);
void LED1_Deinit(void);
```

Functionality

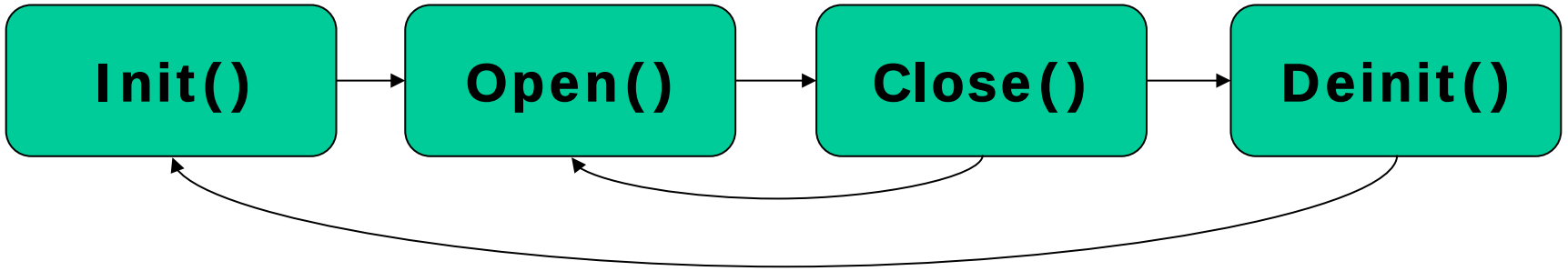
Open & Close

Initialization



Device Driver Flow

- **Init()**
  - Initialization/allocation of memory, data structure, …
- **Open()**
  - Lock device, get device handle, …
- **Close()**
  - Free device, return device handle, …
- **Deinit()**
  - Free memory, reset to default, …



Device Handle

- One Interface for all devices
- Need to pass device information (handle)
  - Flexible
  - Overhead (might be eliminated with macros/inlining)

```
LED_DeviceHndl LED_Init(void);
void LED_On(LED_DeviceHndl led);
void LED_Off(LED_DeviceHndl led);
void LED_Neg(LED_DeviceHndl led);
bool LED_Get(LED_DeviceHndl led);
void LED_Put(LED_DeviceHndl led, bool val);
```

Application Initialization Flow

```
/* main.c */
#include “Application.h"

void main(void) {
  APP_Start();
  for(;;){}
}
```

```
/* Application.c */
void APP_Start(void) {
  PL_Init();
  /* run application */
  PL_Deinit();
}
```

```
/* Platform.c */
void PL_Init(void) {
  LED_Init();
  SCI_Init();
  …
}
```

```
/* Platform.c */
void PL_Deinit(void) {
  …
  SCI_Deinit();
  LED_Init();
}
```

Lab: LED

- Platform.h, Platform_Local.h
  - PL_CONFIG_HAS_LED (0 or 1)
  - PL_CONFIG_NOF_LED (0, 1, ...)
- Functions
  - On, Off, Neg, Get, Put
- Inspect/Verify Functionality