Advanced Microcontroller Laboratory

Summer Term 2018

Introduction Course

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Prof. Dr. Leonhard Reindl, vertreten durch Prof. Dr. Christian Schindelhauer,
Alexander Richter, Sebastian Stöcklin,
Julian Reimer, Elias Rosch

Lehrstuhl für Elektrische Mess- und Prüfverfahren

Part 1

UNI FREIBURG

- Organizational Introduction -

The people…

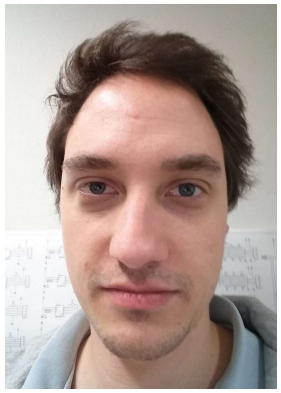
Prof. Dr. Leonhard Reindl

Alexander Richter

Sebastian Stöcklin

Prof. Reindl is taking a sabbatical in this semester. He is represented by Prof. Dr. Christian Schindelhauer.

Julian Reimer



Elias Rosch

Contents I

UNI FREIBURG

Advanced topics in microcontrollers:

- interfacing of *advanced peripherals*
- implementation of *communication interfaces*
- writing *hardware drivers/ libraries* for sensor chips
- accessing *low-power modes*
- writing programs with larger size and complexity

Contents II

UNI FREIBURG

Practical workflow of embedded engineers:

- understanding advanced electronics
- gathering information from data sheets and application notes
- developing solution concepts
- writing structured code
- …

⇒ „real world szenario“

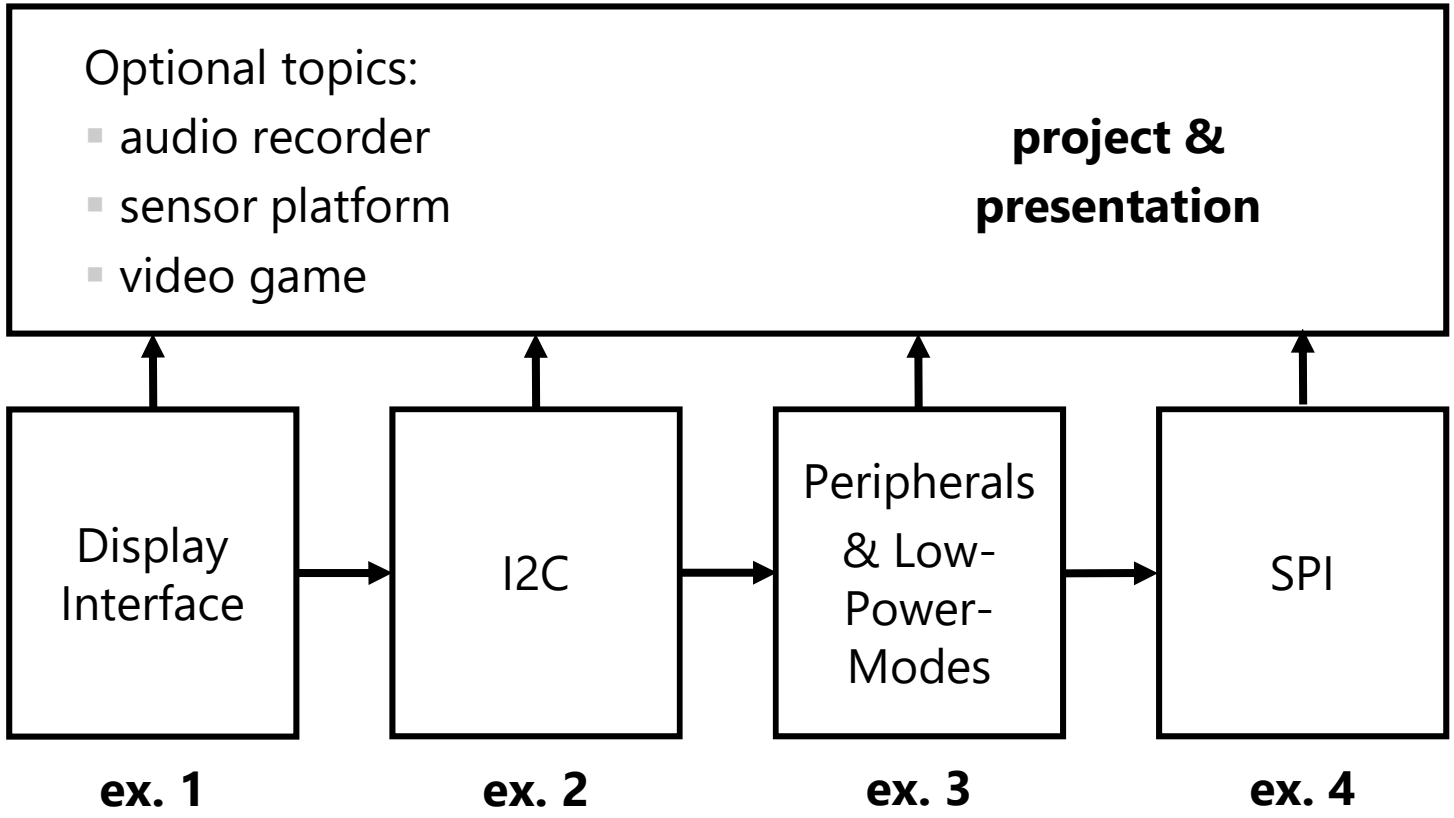
Procedure I

UNI FREIBURG

- no lecture, just practical exercise
- exercises are performed autonomously at home
- online forums for support ⇒ ILIAS platform
- Consultancy hours:
  Wednesday, 16.00 to 18.00, building 106, room 01-007

- 4 exercise sheets
  - 10 points, 2 weeks of time per exercise sheet
  - online submission on ILIAS until Saturday, 23:59
- 1 project at the end of the course
  - 20 points, 4 weeks of time
  - online submission + presentation + colloquium

Procedure II



Grading I

UNI FREIBURG

- Exercises are considered as examination, no exam!
  ⇒ „Prüfungsleistung“
- 3 ECTS
- Grade is based on exercise points
  4 exercises × 10 pts. + 1 project × 20 pts. = 60 pts.
- Please register for the examination!
  Registration period: 01.06.2018 to 13.07.2018

Grading II

UNI FREIBURG

- Distribution of points is transparent:
  - given on exercise sheets for subtasks
  - given in the header files for subroutines

- Criteria to receive points:
  - code is working as expected by the task
  - concept is clear and reasonable
  - code is commented

Plagiarism

UNI FREIBURG

- Remember: exercises = examination!
- In our case:
  plagiarism = strong similarity of code and comments

> A single occurrence of plagiarism will immediately lead to the exclusion from the course and to a grading of 5.0!

- Also note: Multiple plagiarism might lead to the exclusion from the study program.

Requirements I

UNI FREIBURG

- extension to the basic courses
  - „Mikrocomputertechnik“
  - „Microcontroller Techniques“

- Thus, you will require knowledge in the fields of …
  - microcontrollers
  - hardware related C programming
  - electronics

Requirements II

UNI FREIBURG

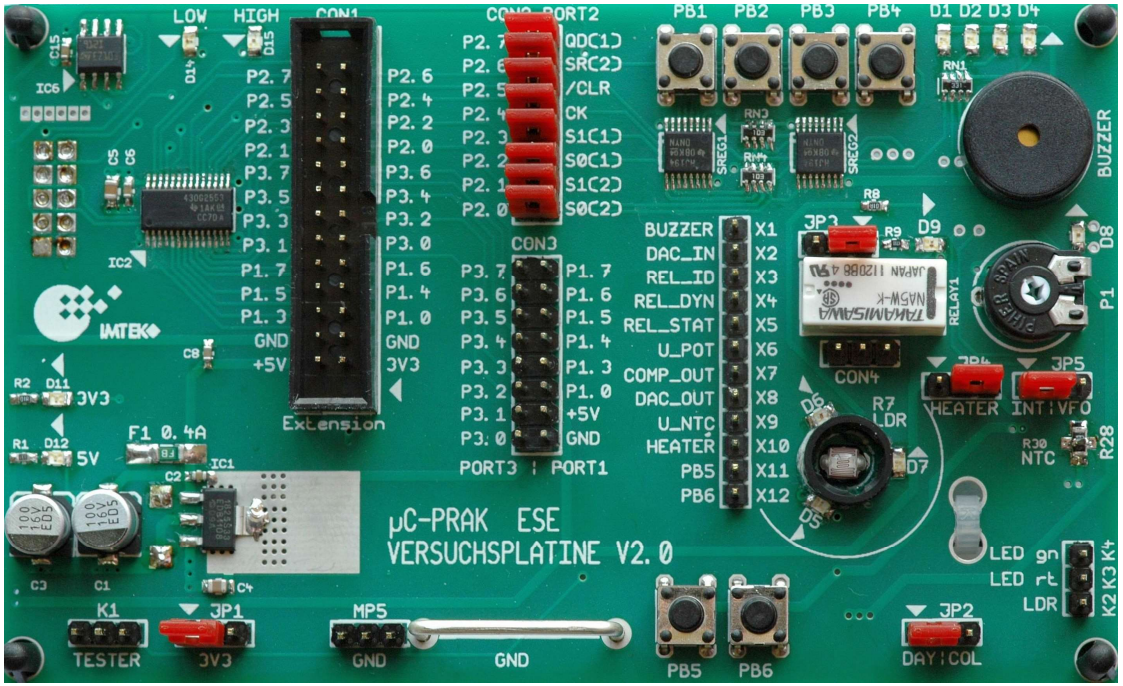
1) Hardware: borrow from TF library, building 101
   - µC-Praktikumskoffer I
   - µC-Praktikumskoffer II

2) Software:
   - Code Composer Studio (IDE): http://processors.wiki.ti.com/index.php/Download_CCS
   - serial terminal (e.g. HTerm): http://www.der-hammer.info/terminal/
   - IKALOGIC ScanaStudio (Logic Analyzer Software): https://ikalogic.com/pages/discontinued-products

Requirements II

1. Hardware box – „µ-Controller-Praktikum 1“

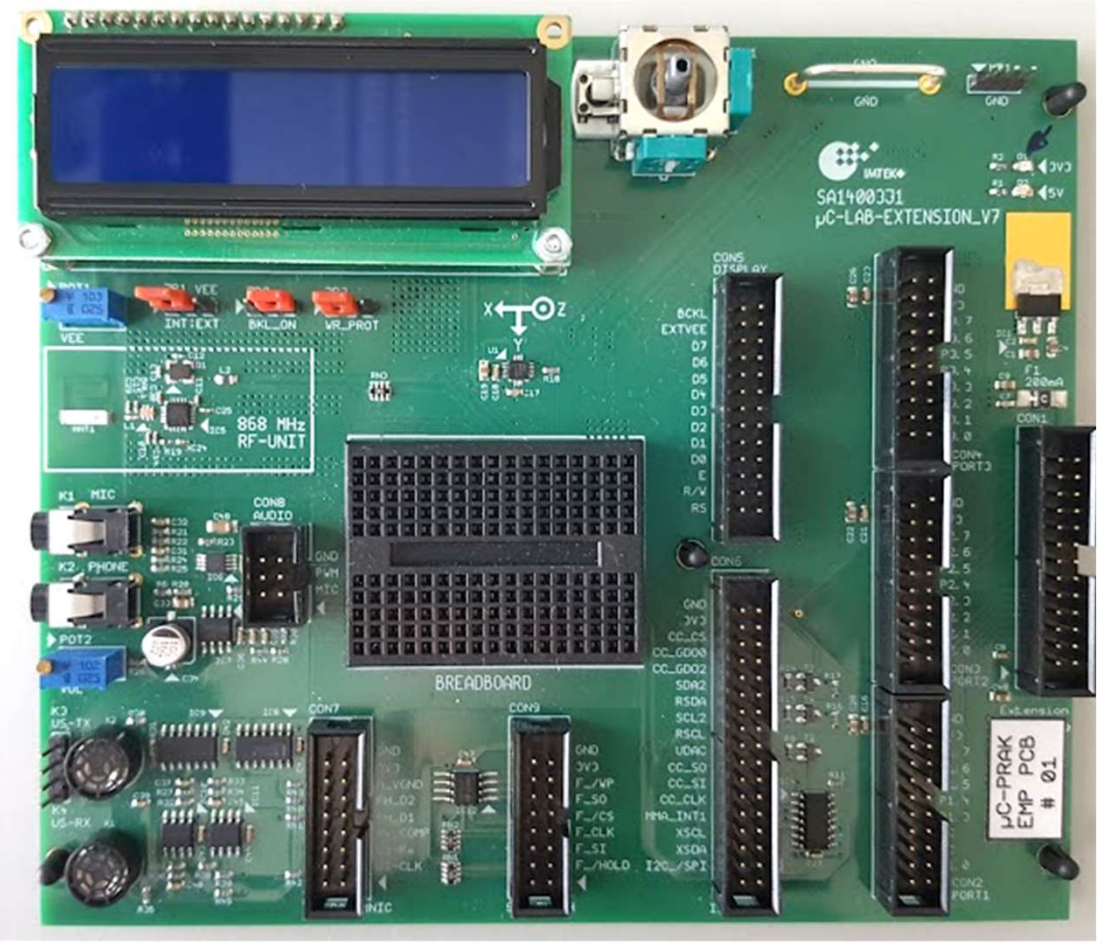
- available in the library of the faculty (building 101)
- Content: controller board



+ USB cable
+ jumper wires
+ plastic tube
+ colored chips
+ logic analyzer

Requirements II

2. Hardware box – „µ-Controller-Praktikum 2“



ILIAS

UNI FREIBURG

- all information is provided on ILIAS:
  - introduction documents
  - exercise sheets & exercise upload
  - datasheets
  - board schematics

- *ilias.uni-freiburg.de* > Magazin
  > Lehrveranstaltungen im SoSe 2018
  > Technische Fakultät
  > Embedded Systems Engineering
  > Master – Wahlbereich
  > Circuits and Systems

Part 2

UNI FREIBURG

- Technical Introduction -

Hardware Debugging

UNI FREIBURG

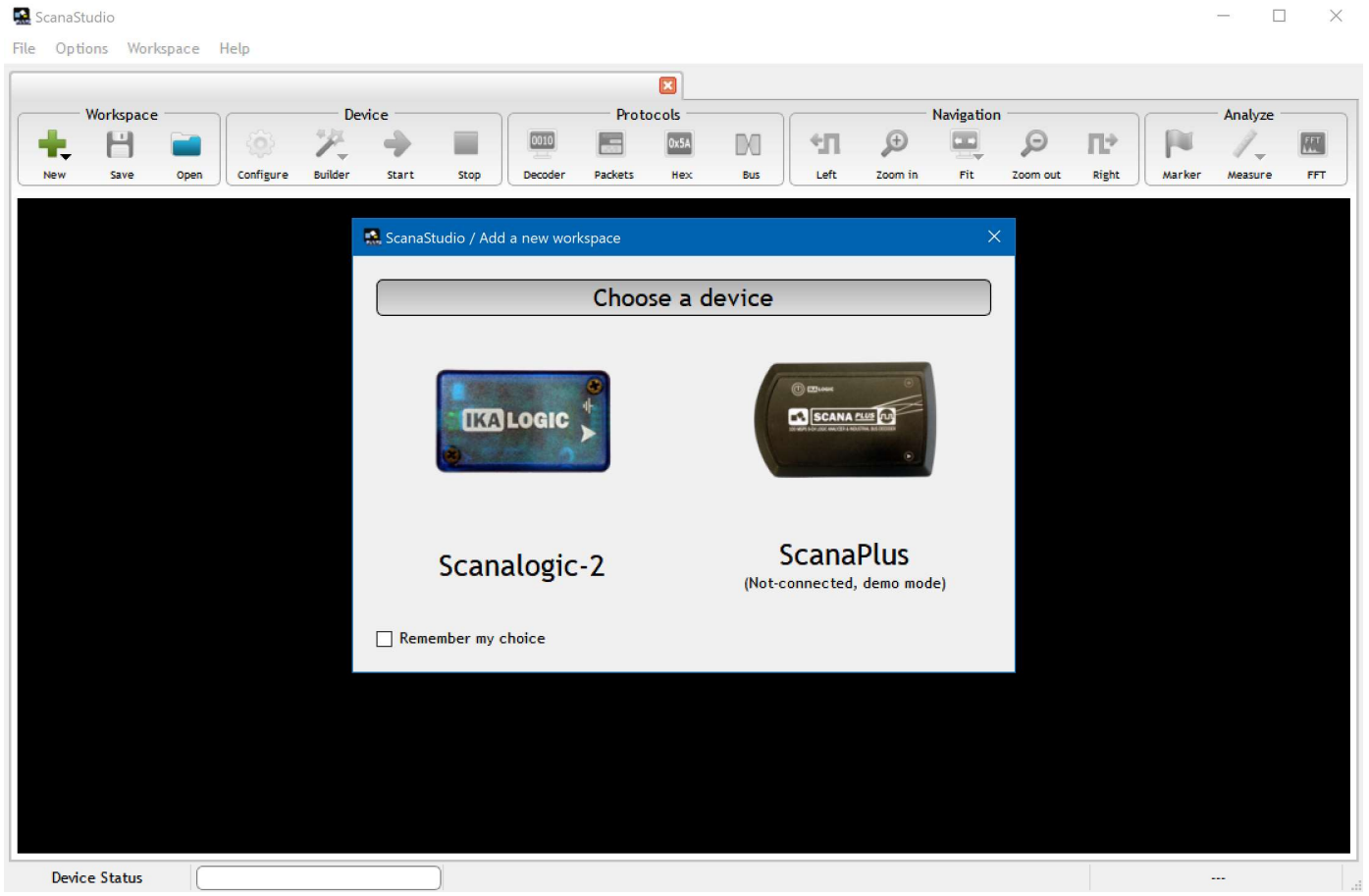
Logic Analyzer

- hardware: IKALOGIC Scanalogic-2
- software: ScanaStudio 2.3

- capture time sequence of digital signals
  - up to 4 channels
  - up to 20 MSPS
  - up to 256,000 samples per channel
- decoding of communication protocolls
- synthesis of digital signals with up to 600 Hz

Hardware Debugging

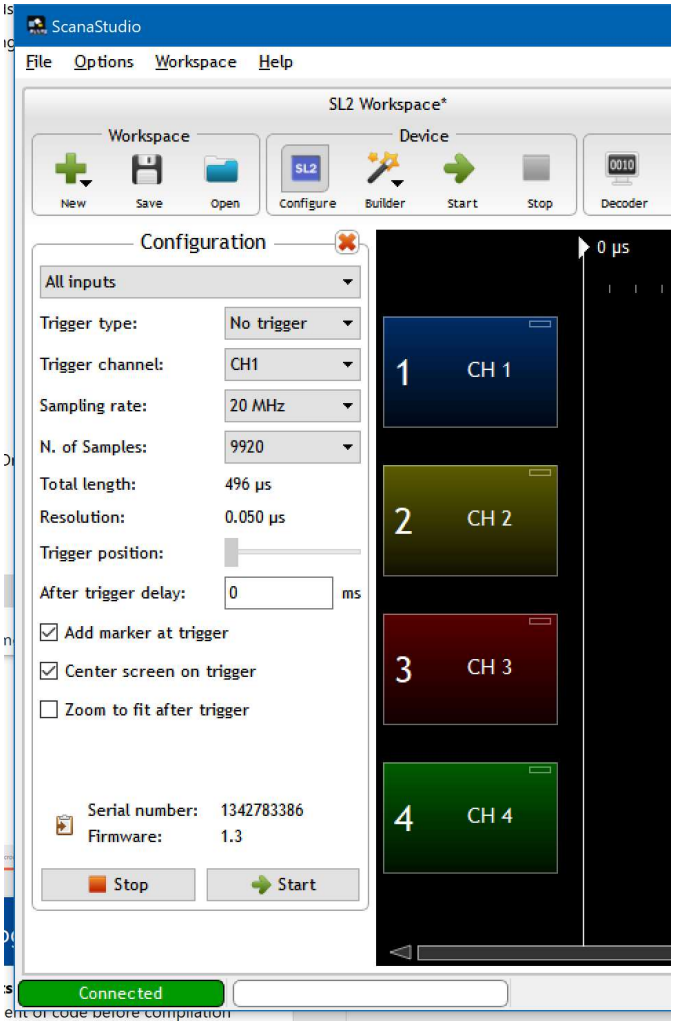
New > Scanalogic-2



Hardware Debugging

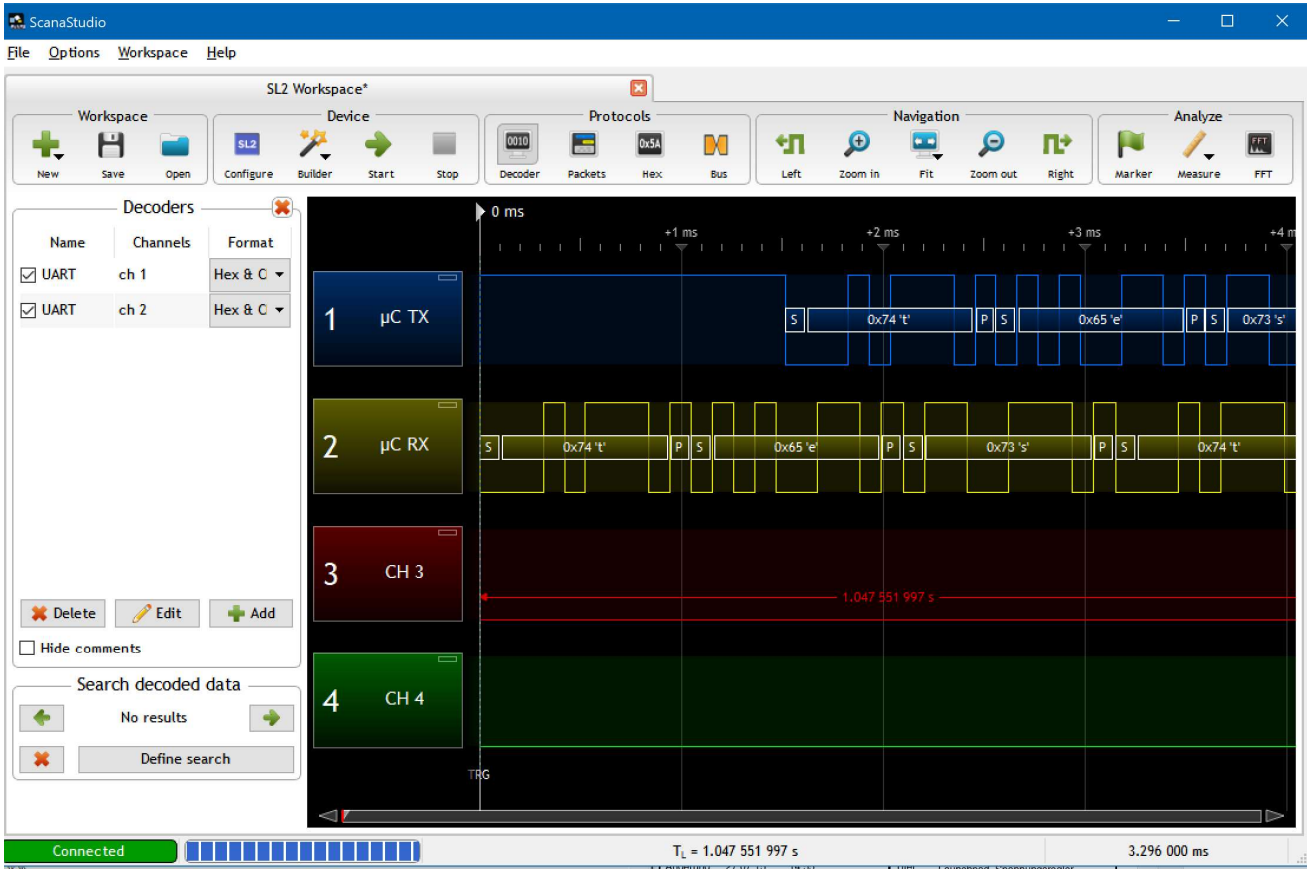
Configuration

- select I/Os
- activate and choose trigger & trigger channel
  start measurements on rising or falling edges
- specify sampling rate and no. of samples
  with effect on the duration
- start measurement



Hardware Debugging

Decoders



Hardware Debugging

UNI FREIBURG

- real-world demo -

Advanced µC Programming

UNI FREIBURG

Preprocessor statements

- performing (re)placement of code before compilation
- constants should be capital letters with underscores

- `#include <math.h>`
  inserts the content of the given file
  `<…>` includes from standard path
  `"folder/file.h"` includes from your project path
  "./file.h"
- `#define MAX_VALUE 5`
  defines a constant; every occurence of this constant will be replaced by the expression (here: by 5).

#define LED_PORT P3OUT
#define LED_PIN BIT0
P3.0
LED_PORT |= LED_PIN;

Advanced µC Programming

UNI FREIBURG

Preprocessor statements

```
#if MAX_VALUE == 5
// this code is compiled for value being 5
#elif MAX_VALUE == 10
// that code is compiled for value being 10
#endif
```

allows to include or exclude code based on defined constants

Advanced µC Programming

UNI FREIBURG

Preprocessor statements

- ```
  #ifdef MAX_VALUE
  // this code is compiled when MAX_VALUE exists
  #endif
  ```

#ifndef

allows to include or exclude code based on the availability of a constant
  ```

Advanced µC Programming

UNI FREIBURG

Functions

- Use them!
- Name them appropriately:

```
// No one knows what is does:
void func1(char * a);
```

versus

```
// Quite clear:
void serialPrint(char * text);
```

Advanced µC Programming

UNI FREIBURG

Functions

- How to hand over an array:

```
void serialPrint(char * text) {
    int i = 0;
    while (text[i] != 0) {
        serialWrite(text[i++]);
    }
}

void main(void) {
    ...
    serialWrite("Hi!");
    ...
}
```

```
char text2[4] = { ... }
serialPrint (text2 );
serialPrint (&text2[0]);
```

Advanced µC Programming

UNI FREIBURG

Libraries

- a set of subroutines and variables
  to handle specific tasks of a similar field
- examples: functions to
  - perform mathematical operations
  - control a display
  - store data in flash memory
  - …
- consist of a *header file* (.h) and a *code file* (.c)

Advanced µC Programming

UNI FREIBURG

Libraries

- *Header file* (.h) = interface to the library
  - contains the function declarations (function prototypes with parameters and return type)
  - contains the definition of parameters & constants
  - included in the C files which use the lib

- *Code file* (.c) = actual implementation
  - contains the function definitions
  - your source code goes here

Advanced µC Programming

UNI FREIBURG

Libraries

```
#ifndef TEMPLATEEMP_H_
#define TEMPLATEEMP_H_

#include <msp430g2553.h>

#define RXBUFFERSIZE 32
extern char rxBufferStart;
void serialWrite(char tx);
void serialPrintInt(int i);

...



#endif /* TEMPLATEEMP_H_ */
```

templateEMP.h

```
#include "libs/templateEMP.h"

char rxBuffer[RXBUFFERSIZE];
char rxBufferStart = 0;
char rxBufferEnd = 0;

void serialWrite(char tx) {
   // Wait for empty TX buf:
   while (!(IFG2 & UCA0TXIFG));
   // Write char to TX.
   UCA0TXBUF = tx;
   // Wait 'til TXed:
   while (!(IFG2 & UCA0TXIFG));
}
...
```

templateEMP.c

Structs

- combine variables to a logical unit

```
// Buffer type definition:
typedef struct {
    char data[BUFFER_SIZE];
    char start;
    char end;
    char error;
} Buffer_t;

// Ring buffer definition:
Buffer_t ringBuffer = {
    .start  = 0,
    .end    = 0,
    .error  = 0,
};
```

```
// Ring buffer usage:
ringBuffer.start = 0;
ringBuffer.end   = 0;

ringBuffer.data[2] = 'a';
```

(ringBuffer.start)++

Advanced µC Programming

UNI FREIBURG

General recommendations:

- To track time, use a timer!
  `__delay_cycles(1000)` might be okay sometimes, but it is neither precise nor nice.

- No magic numbers!
  Numbers that occur multiple times in your code be replaced by constants (see `#define`).

- Consider code examples to get started.

- Auto-formatting in CCS: CTRL + SHIFT + F

Exercise Sheet I

UNI FREIBURG

- check exercise sheet -