

Übungen zu Systemnahe Programmierung in C (SPiC) – Wintersemester 2017/18

Übung 1

Benedict Herzog
Sebastian Maier

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme

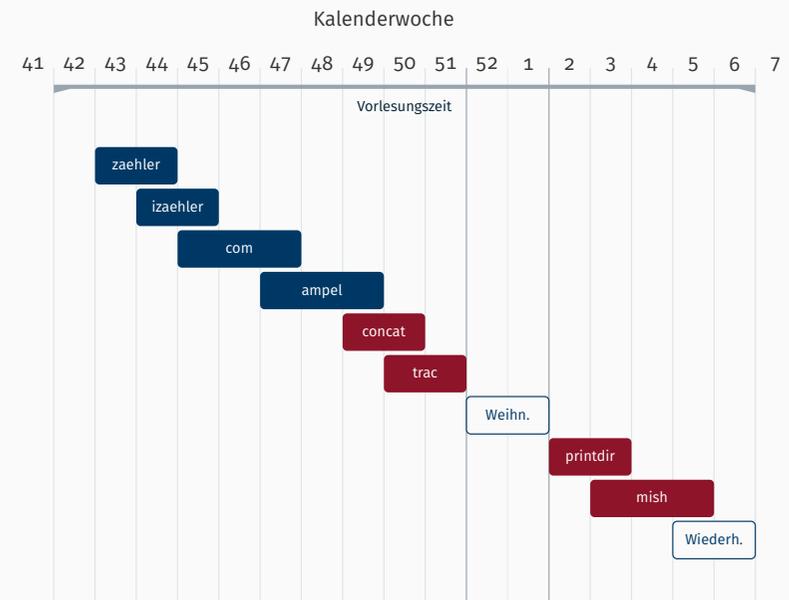


Organisatorisches

Tafelübungen

- Tafelübung Do 10:15 – 11:45 (im Raum 01.153-113)
- Ablauf der Tafelübungen:
 1. Besprechung der alten Aufgabe
 2. Praxisnahe Vertiefung des Vorlesungsstoffes
 3. Vorstellung der neuen Aufgabe
 4. ggf. Entwicklung einer Lösungsskizze der neuen Aufgabe
 5. Hands-on: gemeinsames Programmieren
- Folien nicht unbedingt zum Selbststudium geeignet
→ Anwesenheit, Mitschrift
- Übersicht aller SPiC-Termine:
https://www4.cs.fau.de/Lehre/WS17/V_SPiC/#woch
- Semesterplan:
https://www4.cs.fau.de/Lehre/WS17/V_SPiC/#sem

Aufgaben



- Abgabe unter Linux
- automatische Plagiatsprüfung
 - Vergleich mit allen anderen (auch älteren) Lösungen
 - “abgeschriebene” Lösungen bekommen 0 Punkte⇒ Im Zweifelsfall beim Übungsleiter melden
- Punktabzug
 - -2 Punkte je Compilerwarnung
 - -50% der möglichen Punkte falls nicht übersetzbar
- Kommentare im Code helfen euch und dem Korrektor

3

- abgegebene Aufgaben werden mit Übungspunkten bewertet
- ab 50% der erreichbaren Übungspunkte gibt es Bonuspunkte für die Klausur
- Umrechnung der Übungspunkte in Bonuspunkte für die Klausur (bis zu 10% der Punkte)
- Bestehen der Klausur durch Bonuspunkte *nicht möglich*
- Bonuspunkte nicht in nächste Semester übertragbar

4

- Rechnerübung Fr 14:15 – 15:45 (im Raum 01.153-113)
- Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
- Falls 30 Minuten nach Beginn der Rechnerübung (also um 14:45) niemand anwesend ist, kann der Übungsleiter gehen

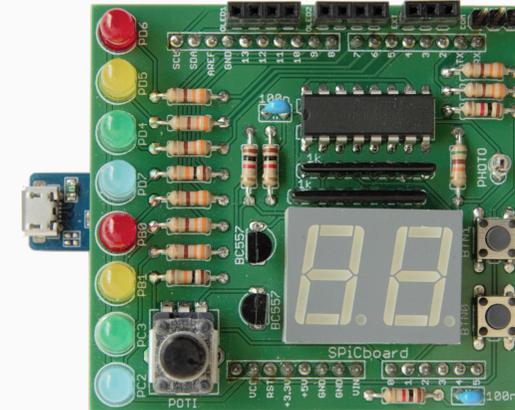
- diese Folien konsultieren
- häufig gestellte Fragen (FAQ) und Antworten:
https://www4.cs.fau.de/Lehre/WS17/V_SPIC/Uebung/faq
- Fragen zu Übungsaufgaben im EEI-Forum posten (darf auch von anderen Studienrichtungen verwendet werden)
<https://eei.fsi.uni-erlangen.de/forum/forum/16>
- bei speziellen Fragen Mail an Mailingliste, die alle Übungsleiter erreicht: i4spic@cs.fau.de
⇒ zum Beispiel auch, wenn kein Übungsleiter auftauchen sollte

5

6

Entwicklungsumgebung

- **ATmega328PB Xplained Mini:**
Mikrocontroller-Board mit integriertem Programmer/Debugger
- Speziell für (G)SPiC angefertigte **SPiCboards** als Erweiterungsplatine



7

Aufgabenbearbeitung

- Betreute Bearbeitung der Aufgaben während der Rechnerübung
 - ⇒ Hardware wird zur Verfügung gestellt
- Selbständige Bearbeitung teilweise nötig
 - eigenes SPiCboard: Anfertigung am Lötabend
 - SPiCboard Simulator: SPiCsim

8

Funktionsbibliothek

- **libspicboard:** Funktionsbibliothek zur Ansteuerung der Hardware
Beispiel: `sb_led_on(GREEN0);` schaltet 1. grüne LED an
- direkte Konfiguration der Hardware durch Anwendungsprogrammierer nicht nötig
- Verwendung vor allem bei den ersten Aufgaben, später muss `libspicboard` teils selbst implementiert werden
- Dokumentation online:
https://www4.cs.fau.de/Lehre/WS17/V_SPIC/Uebung/doc

9

- Projektverzeichnis
 - in Windows: Q:\
 - in Linux: /proj/i4spic/LOGINNAME/
 - Lösungen hier in Unterordnern aufgabeX speichern
 - ⇒ das Abgabeprogramm sucht (nur) dort
 - für andere nicht lesbar
 - wird automatisch erstellt
- Heimverzeichnis
 - in Windows: Z:\
 - in Linux: ~

10

- Vorgabeverzeichnis:
 - Windows: P:\
 - Linux: /proj/i4spic/pub/
 - Projekttemplate SPiCboard_Project_Template.zip für Atmel Studio 7
 - Hilfsmaterial und Binärmusterlösungen zu jeder Übungsaufgabe unter aufgabeX/
- Falls eines der Netzlaufwerke nicht angezeigt wird:
 - Windows Explorer – Computer – Map network drive
 - Z:\ unter \\fai03\LOGINNAME
 - P:\ unter \\fai03\i4spicpub
 - Q:\ unter \\fai03\i4spichome

10

Entwicklungsumgebung: Atmel Studio

- Programmentwicklung mit Atmel Studio 7 unter Windows
- vereint Editor, Compiler und Debugger in einer Umgebung
- Cross-Compiler zur Erzeugung von Programmen für unterschiedliche Architekturen
 - Wirtssystem (engl. host): Intel-PC
 - Zielsystem (engl. target): AVR-Mikrocontroller

Anleitung

11

Windows-Login

- Zur Bearbeitung der Übungen ist ein Windows-Login nötig:
 - Im Raum 01.155 mit Linux-Passwort einloggen
 - Ein Terminalprogramm öffnen und dort folgendes Kommando ausführen:
`cip-set-windows-password`
- Kriterien für sicheres Passwort:
 - Mindestens 8 Zeichen, besser 10
 - Mindestens 3 Zeichensorten, besser 4 (Groß-, Kleinbuchstaben, Zahlen, Sonderzeichen)
 - Keine Wörterbuchwörter, Namen, Login, etc.
- Passwort-Generierung zum Ausschuchen mit folgendem Kommando:
`pwgen -s 12`

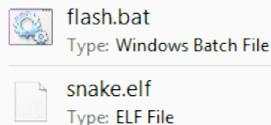
12

Demo

13

Binärabbild flashen

- Nötig, um vorgefertigte Binärabbilder (.elf-Images) zu testen, z. B. Binärmusterlösungen unter P:\aufgabeX
- Möglich mit Skript `flash.bat` im jeweiligen Verzeichnis, Ausführen mit Doppelklick



- Nach erfolgreichem Flashen führt das Board das Programm direkt aus
- Neustart des Programms ist durch Trennen und Wiederherstellen der USB-Stromversorgung möglich

14

Abgabe (1)

- Spätestens nach erfolgreichem Testen des Programms müssen Übungslösungen zur Bewertung abgegeben werden
- **Bei Zweiergruppen darf nur ein Partner abgeben!**
- Abgabe unter einer Linux-Umgebung per Remote Login
 - Start ~> Alle Programme ~> PuTTY ~> PuTTY
 - Host Name: `faii0sr0` bzw. von Zuhause `faii0sr0.cs.fau.de`
 - Open
 - PuTTY Security Alert mit "Ja" bestätigen
 - Login mit Benutzernamen und **Linux**-Passwort
- Im erscheinenden Terminal-Fenster folgendes Kommando ausführen (aufgabeX entsprechend ersetzen):
`/proj/i4spic/bin/submit aufgabeX`
- Wichtig: **Grüner Text** signalisiert erfolgreiche Abgabe, **roter Text** einen Fehler!

15

- Fehlerursachen
 - Notwendige Dateien liegen nicht im richtigen Ordner
 - aufgabeX muss klein geschrieben sein
 - .c-Datei falsch benannt

- Nützliche Tools
 - Quelltext der abgegebenen Aufgabe anzeigen:
/proj/i4spic/bin/show-submission aufgabeX
 - Unterschiede zwischen abgegebener Version und Version im Projektverzeichnis Q:\aufgabeX anzeigen:
/proj/i4spic/bin/show-submission aufgabeX -d
 - Eigenen Abgabetermin anzeigen:
/proj/i4spic/bin/get-deadline aufgabeX

16

Variablen

Verwendung von int

- Die Größe von int ist nicht genau definiert
 - ⇒ Gerade auf µC führt dies zu Fehlern und/oder langsameren Code
 - Für die Übung gilt
 - Verwendung von int ist ein "Fehler"
 - Stattdessen: Verwendung der in der stdint.h definierten Typen: int8_t, uint8_t, int16_t, uint16_t, etc.
 - Wertebereich
 - limits.h: INT8_MAX, INT8_MIN, ...
 - Speicherplatz ist sehr teuer auf µC
- ~> Nur so viel Speicher verwenden, wie tatsächlich benötigt wird!

Sichtbarkeit & Lebensdauer

Sichtbarkeit und Lebensdauer	nicht static	static
lokale Variable	Sb Block Ld Block	Sb Block Ld Programm
globale Variable	Sb Programm Ld Programm	Sb Modul Ld Programm
Funktion	Sb Programm	Sb Modul

- Lokale Variable, nicht static = auto Variable
- ~> automatisch allokiert & freigegeben
- Funktionen als static, wenn kein Export notwendig

```

01 static uint8_t state; // global static
02 uint8_t event_counter; // global
03
04 void main(void) {
05     /* ... */
06 }
07
08 static void f(uint8_t a) {
09     static uint8_t call_counter = 0; // local static
10     uint8_t num_leds; // local (auto)
11     /* ... */
12 }
    
```

- Sichtbarkeit/Gültigkeit möglichst weit **einschränken**
- Globale Variable ≠ lokale Variable in f()
- Globale static Variablen: Sichtbarkeit auf Modul beschränken

Aufgabe: Zähler

```

01 #define PB3 3
02 typedef enum {
03     BUTTON0 = 0,
04     BUTTON1 = 1
05 } BUTTON;
06
07 void main(void) {
08     /* ... */
09     PORTB |= (1 << PB3); // nicht (1 << 3)
10
11     BUTTONSTATE old, new; // nicht uint8_t old, new;
12
13     // Deklaration: BUTTONSTATE sb_button_getState(BUTTON btn);
14     old = sb_button_getState(BUTTON0); // nicht sb_button_getState(0)
15     /* ... */
16 }
    
```

- Vordefinierte Typen verwenden
- Explizite Zahlenwerte nur verwenden, wenn notwendig

19

20

Aufgabenbeschreibung: Zähler

- Automatisches Hochzählen mit einer über das Potentiometer einstellbare Geschwindigkeit
- Anzeige erfolgt über 7-Segmentanzeige und LEDs
 - LEDs zu Beginn aus
 - Hunderterstelle durch LED visualisieren: LED0 ≙ 100, LED1 ≙ 200, etc.
 - bei Verlassen des anzeigbaren Wertebereichs Zähler zurücksetzen
- Bibliotheksfunktionen für Potentiometer, 7-Segmentanzeige und LED – Dokumentation auf der Webseite unter https://www4.cs.fau.de/Lehre/WS17/V_SPIC/Uebung/doc

21

- Modulo ist der Divisionsrest einer Ganzzahldivision
- **Achtung:** In C ist das Ergebnis im negativen Bereich auch negativ
- Beispiel:

```
b = a % 4;
```

```
a = -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6
b = -1 | 0 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 0 | 1 | 2
```

22

Hands-on: Signallampe

- Morsesignale über LED o ausgeben
- Steuerung über Taster 1
- Nutzung der Bibliotheksfunktionen für Button und LED
- Dokumentation der Bibliothek: https://www4.cs.fau.de/Lehre/WS17/V_SPIC/Uebung/doc
- Optional: Implementierung ohne Nutzung der Bibliothek

23

Hands-on: Signallampe
