

# Übungen zu Systemnahe Programmierung in C (SPiC)

Peter Wägemann, Moritz Strübe  
(Lehrstuhl Informatik 4)



Wintersemester 2014/2015



Organisatorisches

Tafelübungen

Rechnerübungen

Bonuspunkte

Bei Problemen

Entwicklungsumgebung

Hardware

Bibliothek

Verzeichnisse

Anleitung

CIP Login

AVR-Studio einrichten

Projekt Anlegen

Flashen

Debuggen

Abgeben



- Tafelübung Do 10:15 – 11:45:

- Besprechung der alten Aufgabe, Hinweis auf häufig gemachte Fehler
- Keine Anwesenheitspflicht; bei unentschuldigter Abwesenheit ggf. 0 Punkte auf die Aufgabe
- Ebenfalls 0 Punkte bei "abgeschriebenen" Lösungen; Achtung: Abgabe von Lösungen aus dem letzten Semester ist heikel.
- Vorstellung der neuen Aufgabe, ggf. gemeinsame Entwicklung einer Lösungsskizze



- Reine Rechnerübung Mi 8:30 – 10:00:
  - Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
  - Falls 30 Minuten nach Beginn der Rechnerübung (also um 9:00) niemand anwesend ist, kann der Übungsleiter gehen.



- Bonuspunkte:

- Abgegebene Aufgaben werden bepunktet
- Umrechnung in Bonus für die Klausur (bis zu 10% der Punkte oder 0,7 Notenpunkte)
- *Bestehen* der Klausur durch Bonuspunkte nicht möglich
- Bonuspunkte oder -note gibt es ab der Hälfte der erreichbaren Übungspunkte



- Diese Folien konsultieren  
⇒ Die Folien sind kein Skript!
- Häufig gestellte Fragen (FAQ) und Antworten:  
[https://www4.cs.fau.de/Lehre/WS14/V\\_SPIC/Uebung/faq.shtml](https://www4.cs.fau.de/Lehre/WS14/V_SPIC/Uebung/faq.shtml)
- 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 Übungleiter auftaucht



- Ausleihe von SPiCboard, Kabeln und Programmierer/Debugger tagsüber möglich:
  - Bei Harald Junggunst, Büro 0.046 (Erdgeschoss RRZE-Gebäude)
  - Übliche Bürozeiten: von 8:00 bis 15:00
  - <https://www4.cs.fau.de/~jungguns/>
- In 01.155N befinden sich weitere Windows-Rechner



- libspicboard: Funktionsbibliothek zur einfachen Ansteuerung der Hardware
- Dokumentation online:  
[https://www4.cs.fau.de/Lehre/WS14/V\\_SPiC/Uebung/doc](https://www4.cs.fau.de/Lehre/WS14/V_SPiC/Uebung/doc)



- Heimverzeichnis:
  - Linux: ~
  - Windows: H:\
- Projektverzeichnis:
  - Linux: /proj/i4spic/LOGINNAME/
  - Windows: P:\
  - Die Lösungen müssen im Unterordner aufgabeX gespeichert werden
    - ⇒ Das Abgabeprogramm sucht dort
  - Ist durch andere nicht lesbar
  - Wird automatisch erstellt



- Vorgabeverzeichnis:
  - Linux: /proj/i4spic/pub/
  - Windows: Q:\
  - Aufgabenstellungen unter aufgaben/
  - Hilfsmaterial und Binärmusterlösungen zu einzelnen Übungsaufgaben unter aufgabeX/
  - Programm zum Testen der Einheiten auf den Boards unter boardtest/
  - libspicboard-Bibliothek und -Dokumentation unter i4/
  - Kleine Hilfsprogramme unter tools/
- Falls eines der Verzeichnisse H:\, P:\, Q:\ nicht angezeigt wird:
  - Windows Explorer – Computer – Netzlaufwerk verbinden
  - H:\ unter \\faui03\LOGINNAME
  - P:\ unter \\faui03\i4spichome
  - Q:\ unter \\faui03\i4spicpub



- Zur Bearbeitung der Übungen ist ein Windows-Login nötig
  - Auf einem CIP-Rechner mit Linux-Passwort einloggen
  - Ein Terminalprogramm öffnen und dort folgendes Kommando ausführen:  
`/local/ciptools/bin/setsambapw`  
(hängt auch auf einem Zettel auf der Wand zum Raum 01.155-N)
- Kriterien für sicheres Passwort:
  - Mindestens 8 Zeichen, besser 10
  - Mindestens 3 Zeichensorten, besser 4 (Großbuchstaben, Kleinbuchstaben, Zahlen, Zeichen)
  - Keine Wörterbuch-Wörter, Namen, Login etc.
- Passwort-Generierung zum Aussuchen mit folgendem Kommando:  
`pwgen -s 12`



- Achtung: Die Anleitung muss **genau** beachtet werden.
  - Start von AVR Studio über: Start  $\rightsquigarrow$  Alle Programme  $\rightsquigarrow$  Atmel AVR Tools  $\rightsquigarrow$  AVR Studio 5.1
  - Falls Windows-Firewall einige Funktionen blockiert, auf “Abbrechen” klicken
  - Importieren der Projektvorlage (einmalig):
    - File  $\rightsquigarrow$  Import  $\rightsquigarrow$  Project Template...
    - Q:\tools\SPiC\_Template5.zip
    - Add to folder: <Root>
    - OK
- $\Rightarrow$  Successfully imported project template



- Pro Übungsaufgabe ein neues Projekt anlegen:
  - File  $\rightsquigarrow$  New  $\rightsquigarrow$  Project...
  - Projekttyp: (G)SPiC-Projekt
  - Name: aufgabeX, zum Beispiel aufgabe0 (Achtung: Kleinschreibung!)
  - Location: P:\
  - **Wichtig:** Kein Häkchen bei "Create directory for solution"
  - OK
  
- Initiale C-Datei zu Projekt hinzufügen:
  - Rechts Solution Explorer auswählen und dort orangefarbenes Projekt auswählen
  - Project  $\rightsquigarrow$  Add New Item...
  - Dateityp: C File
  - Name: siehe Aufgabenstellung, jetzt test.c (Achtung: Kleinschreibung!)
  - Add



- Beispielprogramm, um erste grüne LED einzuschalten:

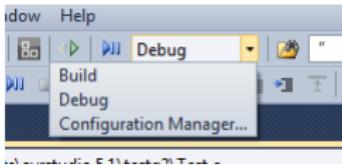
```
1 #include <led.h>
2
3 void main(void) {
4     sb_led_on(GREEN0);
5     while(1) { /* Endlosschleife */
6     }
7 }
```

- Programm kompilieren mit Build ↗ Build Solution
  - ⇒ Programm wurde nur erfolgreich übersetzt, wenn unten steht: Build succeeded.
  - ⇒ Fehlermeldungen erscheinen ggf. unten



- *Achtung:* Zwei verschiedene Compiler-Profile:
  - Debug: Ohne Optimierung
  - Build: Mit Optimierung

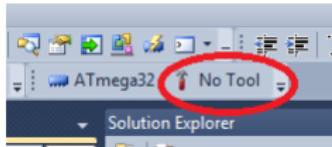
⇒ Optimierung macht den Code *sehr* viel schneller, kann aber den Debugger "verwirren".
- Umstellung des Profils in Drop-Down-Box rechts neben dem Play-Button in der Werkzeugeiste



- Letztendlich soll jede Aufgabe mit Build kompiliert und getestet werden

⇒ *Die Build-Konfiguration wird von uns bewertet!*

- Flashen: Kompiliertes Programm in den Speicher des Mikrocontrollers kopieren
- Programmierer auswählen:
  - Project  $\rightsquigarrow$  aufgabeX Properties
  - Tool  $\rightsquigarrow$  Selected Debugger  $\rightsquigarrow$  AVRISP mkII
  - ISP Clock: 150,00 kHz
  - File  $\rightsquigarrow$  Save Selected Items (CTRL-S)
- Schnellauswahl das Werkzeugs:



- Übersetzen, in den Speicher kopieren und laufen lassen: Debug  $\rightsquigarrow$  Continue (F5)
- (Beim ersten Mal ggf. Firmware-Upgrade durchführen lassen.)



- JTAG-Debugger zum Untersuchen des Programmablaufs “live” auf dem Board
- Debugger auswählen:
  - Project  $\rightsquigarrow$  aufgabeX Properties
  - Tool  $\rightsquigarrow$  Selected Debugger  $\rightsquigarrow$  JTAGICE mkII
  - JTAG Clock: 200,00 kHz
  - File  $\rightsquigarrow$  Save Selected Items
- Direkt in den Speicher kopieren und laufen lassen: Debug  $\rightsquigarrow$  Continue (F5)
- Beim ersten Mal ggf. Firmware-Upgrade durchführen lassen.
- Sollte sich der Debugger eigenartig verhalten ist wahrscheinlich die Clock verstellt.



- Programm laden und beim Betreten von `main()` anhalten: Debug ↗  
Start Debugging and Break
- Schrittweise abarbeiten mit
  - F10 (Step Over): Funktionsaufrufe werden in einem Schritt bearbeitet
  - F11 (Step Into): Bei Funktionsaufrufen wird die Funktion betreten
- Debug ↗ Windows ↗ I/O View: I/O-Ansicht gibt Einblick in die Zustände der I/O-Register; die Werte können dort auch direkt geändert werden
- Breakpoints unterbrechen das Programm einer bestimmten Stelle
  - Setzen durch Codezeile anklicken, dann F9 oder Debug ↗ Toggle Breakpoint
  - Programm laufen lassen (F5 oder Debug ↗ Continue): stoppt, wenn ein Breakpoint erreicht wird



- Nötig, um vorgefertigte Binärabbilder (.hex-Images) zu testen, z. B. Binärmusterlösungen unter Q:\aufgabeX
- Möglich mit Debugger (ICE) oder Programmierer (ISP)
  - Tools ↗ AVR Programming
  - Tool: JTAGICE mkII bzw. AVRISP mkII
  - Device: ATmega32
  - Interface: JTAG bzw. ISP
  - Apply
  - Verbindung überprüfen mit Device ID – Read
    - ↗ Ergebnis: 0x1E 0x95 0x02
      - ⇒ Eignet sich gut um schnell die Verbindung zwischen PC und µC zu testen
  - Memories ↗ Flash: .hex-Datei auswählen
  - Program
- Nach erfolgreichem Flashen führt das Board das Programm direkt aus
- Ein Neustart des Programms ist durch Trennung und Wiederherstellung der USB-Stromversorgung möglich



- Nach erfolgreichem Testen des Programms müssen Übungslösungen zur Bewertung abgegeben werden
- Wichtig: Bei Zweiergruppen darf nur ein Partner abgeben!
- Die Abgabe erfolgt unter einer Linux-Umgebung per Remote Login:
  - Start ↗ Alle Programme ↗ PuTTY ↗ PuTTY
  - Host Name: faui0sr0 bzw. von Zuhause faui0sr0.cs.fau.de
  - Open
  - PuTTY Security Alert mit "Ja" bestätigen
  - Login mit Benutzername und **Linux**-Passwort
- Im erscheinenden Terminal-Fenster folgendes Kommando ausführen, dabei **aufgabe0** entsprechend ersetzen:  
`/proj/i4spic/bin/submit aufgabe0`
- Wichtig: **Grüner Text** signalisiert erfolgreiche Abgabe, **roter Text** einen Fehler!



- Fehlerursachen
  - `aufgabeX` muss klein geschrieben sein
  - Häkchen bei "Create directory for solution" nicht weg gemacht:  
⇒ Dateien sind im Ordner `aufgabeX/aufgabeX`
  - .c-Datei falsch benannt.
- Anzeige der abgegebenen Aufgabe

```
/proj/i4spic/bin/show-submission aufgabe0
```

  - Zeigt abgegebene Version an
  - Zeigt ggf. Unterschied zwischen abgegebener Version und Version im Projektverzeichnis `P:\aufgabeX` an

