

Übungen zu Systemnahe Programmierung in C (SPiC)

Sebastian Maier
(Lehrstuhl Informatik 4)

Übung 1



Sommersemester 2016



Inhalt

Organisatorisches

Entwicklungsumgebung

Anleitung

Aufgabe 1 & Hands-on



Organisatorisches

Tafelübungen

Aufgaben

Rechnerübungen

Bei Problemen

Entwicklungsumgebung

Anleitung

Aufgabe 1 & Hands-on



Tafelübungen

- 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/SS16/V_SPiC/#woch
- Semesterplan:
https://www4.cs.fau.de/Lehre/SS16/V_SPiC/#sem



- 8 Aufgaben:
 - 5 x Mikrocontroller (SPiCboard)
 - 3 x Linux
- Lösungen:
 - Abgabe unter Linux
 - Lösung wird automatisch auf Ähnlichkeit mit allen anderen, auch älteren Lösungen verglichen
 - "abgeschriebene" Lösungen bekommen 0 Punkte
 - ⇒ Im Zweifelsfall bei einem Übungsleiter melden
 - Programm nicht übersetzbare: -50% der möglichen Punkte
 - Bei Warnungen des Compilers: Je Warnung -2 Punkte
 - Kommentare im Code helfen euch und dem Korrektor
 - Nur die Aufgabenstellung lösen ↗ Code auskommentieren
 - Lieber Teilaufgaben richtig, als alles, aber falsch lösen



Bonuspunkte

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



- Räume der Rechnerübungen: 01.153-113 (und 01.155N-113)
- Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
Freie Plätze nach dem „First come, first served“-Prinzip
- Falls 30 Minuten nach Beginn der Rechnerübung niemand anwesend ist, kann der Übungsleiter gehen
- Termine:
https://www4.cs.fau.de/Lehre/SS16/V_SPIC/#woch



Bei Problemen

- Diese Folien konsultieren
- Häufig gestellte Fragen (FAQ) und Antworten:
https://www4.cs.fau.de/Lehre/SS16/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



Organisatorisches

Entwicklungsumgebung

Hardware

Funktionsbibliothek

Wichtige Verzeichnisse

Atmel Studio

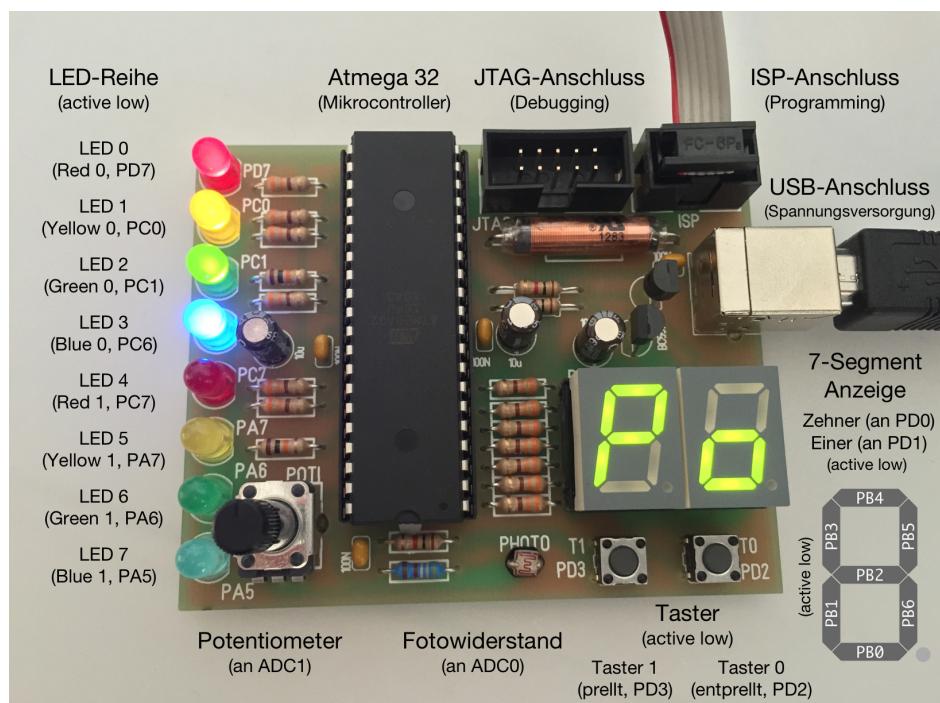
Anleitung

Aufgabe 1 & Hands-on



Hardware: SPiCboard

- Speziell für (G)SPiC angefertigte SPiCboards mit AVR-ATmega32-Mikrocontroller



- JTAG-Debugger (links) zur Überwachung der Programmausführung direkt auf dem Board (z. B. Schritt-für-Schritt-Ausführung, Untersuchung von Variablenwerten, etc.)
- ISP-Programmierer (rechts) zur Übertragung des eigenen Programms auf den Mikrocontroller



Aufgabenbearbeitung

- Betreute Bearbeitung der Aufgaben während der Rechnerübungen
⇒ Hardware wird zur Verfügung gestellt
- Selbständige Bearbeitung teilweise nötig
- 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/>

- 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/SS16/V_SPIC/Uebung/doc



Wichtige Verzeichnisse (1)

- Projektverzeichnis:
 - Windows: P:\
 - Linux: /proj/i4spic/LOGINNAME/
 - Lösungen müssen hier in Unterordnern aufgabeX gespeichert werden
 - ⇒ Das Abgabeprogramm sucht dort
 - Ist durch andere nicht lesbar
 - Wird automatisch erstellt
- Heimverzeichnis:
 - Windows: Z:\
 - Linux: ~



- Vorgabeverzeichnis:
 - Windows: Q:\
 - Linux: /proj/i4spic/pub/
 - Projektvorlage vorlage.cproj für Atmel Studio
 - 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 libspicboard/
 - Kleine Hilfsprogramme unter tools/
- Falls eines der Verzeichnisse Z:\, P:\, Q:\ nicht angezeigt wird:
 - Windows Explorer – Computer – Map network drive
 - Z:\ unter \\faui03\LOGINNAME
 - P:\ unter \\faui03\i4spichome
 - Q:\ unter \\faui03\i4spicpub



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



Organisatorisches

Entwicklungsumgebung

Anleitung

CIP Login

Projekt und Dateien anlegen

Programmieren und Übersetzen

Flashen

Debuggen

Abgabe

Aufgabe 1 & Hands-on



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 Aussuchen mit folgendem Kommando:
`pwgen -s 12`



- Projektordner erstellen und Vorlage kopieren:
 1. Ordner für aktuelle Aufgabe erstellen
P:\aufgabeX\
 2. Die Datei
Q:\vorlage.cproj
in das Verzeichnis der aktuellen Aufgabe kopieren
P:\aufgabeX\
 3. Namen der Projektdatei von vorlage.cproj in aufgabeX.cproj ändern
- Projekt in Atmel Studio öffnen:
 1. Start von Atmel Studio über:
Start – All Programs – Atmel Studio 7.0 – Atmel Studio 7.0
 2. Beim ersten Start öffnet sich ein Dialog, in dem “Standard” als User Interface ausgewählt werden sollte
 3. File – Open → Project/Solution...
 4. P:\aufgabeX\aufgabeX.cproj auswählen
 5. OK



C-Datei hinzufügen

- Initiale C-Datei zu Projekt hinzufügen:
 1. Rechts “Solution Explorer” auswählen und dort orangefarbenes Projekt auswählen
 2. Project – Add New Item...
 3. Dateityp: C File
 4. Name: siehe Aufgabenstellung, jetzt test.c
(Achtung: Kleinschreibung!)
 5. Add



Programmieren und Übersetzen (1)

- Beispielprogramm, um erste grüne LED einzuschalten:

```
1 #include <led.h>
2
3 void main(void) {
4     sb_led_on(GREEN0);
5     while(1) {
6         /* Endlosschleife:
7             Mikrocontrollerprogramm darf nie terminieren */
8     }
9 }
```

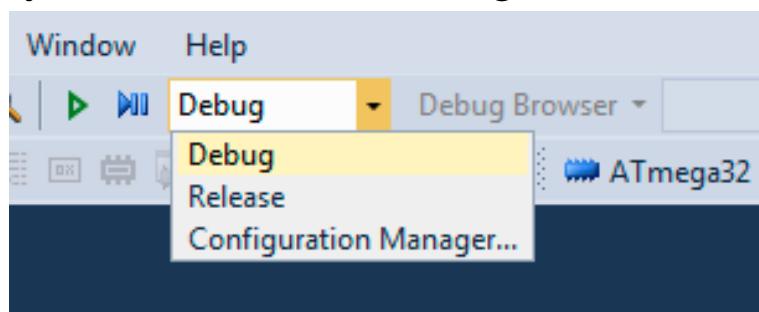
- Programm kompilieren mit Build ↠ Build Solution

⇒ Programm wurde nur erfolgreich übersetzt, wenn unten steht:
Build succeeded.
⇒ Fehlermeldungen erscheinen ggf. unten



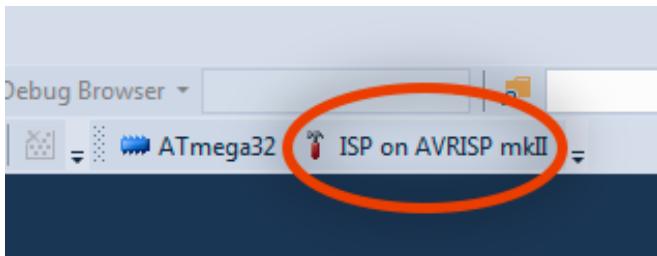
Programmieren und Übersetzen (2)

- Achtung: Zwei verschiedene Compiler-Profile:
 - Debug: Ohne Optimierung
 - Release: 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 Release kompiliert und getestet werden
⇒ Die Release-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

Debuggen (1)

- 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
- Sollte sich der Debugger eigenartig verhalten ist wahrscheinlich die Clock verstellt

- Programm laden und beim Betreten von `main()` anhalten:
Debug \rightsquigarrow 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 \rightsquigarrow Windows \rightsquigarrow 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 an einer bestimmten Stelle
 - Setzen durch Codezeile anklicken, dann F9 oder Debug \rightsquigarrow Toggle Breakpoint
 - Programm laufen lassen (F5 oder Debug \rightsquigarrow Continue): stoppt, wenn ein Breakpoint erreicht wird



Binärabbild flashen

- Nötig, um vorgefertigte Binärabbilder (.hex-Images) zu testen, z. B. Binärmusterlösungen unter Q:\aufgabeX
- Möglich mit Debugger (ICE) oder Programmierer (ISP)
 - Tools \rightsquigarrow Device Programming
 - Tool: JTAGICE mkII bzw. AVRISP mkII
 - Device: ATmega32
 - Interface: JTAG bzw. ISP
 - Apply
 - Verbindung überprüfen mit Device Signature – Read
 - \rightsquigarrow Ergebnis: 0x1E9502
 - \Rightarrow Eignet sich gut um die Verbindung zwischen PC und µC zu testen
 - Memories \rightsquigarrow 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-Spannungsversorgung möglich



Abgabe (1)

- 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 `aufgabeX` entsprechend ersetzen:
`/proj/i4spic/bin/submit aufgabeX`
- Wichtig: **Grüner Text** signalisiert erfolgreiche Abgabe, **roter Text** einen Fehler!



Abgabe (2)

- Fehlerursachen
 - Notwendige Dateien liegen nicht im richtigen Ordner
 - `aufgabeX` muss klein geschrieben sein
 - .c-Datei falsch benannt
- Weitere nützliche Tools:
 - Quelltext der abgegebenen Aufgabe anzeigen:
`/proj/i4spic/bin/show-submission aufgabeX`
 - Unterschiede zwischen abgegebener Version und Version im Projektverzeichnis P:\aufgabeX anzeigen:
`/proj/i4spic/bin/show-submission aufgabeX -d`
 - Eigenen Abgabetermin anzeigen:
`/proj/i4spic/bin/get-deadline aufgabeX`



Organisatorisches

Entwicklungsumgebung

Anleitung

Aufgabe 1 & Hands-on

Aufgabenbeschreibung: Blink
Hands-on: Licht



Aufgabenbeschreibung: Blink

- Lernziel:
 - Umgang mit Programmierwerkzeugen
 - Aktives Warten
- Blinkende LED GREEN1
 - Frequenz ca. 1 mal pro Sekunde
 - Nutzung der Bibliothesfunktionen für LEDs
 - Implementierung durch aktives Warten (Schleife mit Zähler)
 - Übersetzung in Compiler-Konfiguration Release
- Dokumentation der Bibliothek:
https://www4.cs.fau.de/Lehre/SS16/V_SPiC/Uebung/doc
- Abzugebende Datei: blink.c



- Projekt aufgabe0 erstellen
 - Ordner erstellen
 - Projektvorlage kopieren und Dateinamen anpassen
 - Projekt in Atmel Studio öffnen
- Minimalprogramm test.c erstellen
 - schaltet LED GREEN0 ein
 - wartet dann endlos
- Programm übersetzen und im Simulator testen
- Lösung abgeben
 - mit Putty zu Linux-Rechner verbinden
 - Abgabeprogramm ausführen

