

Übungen zu Systemnahe Programmierung in C (SPiC)

Moritz Strübe, Rainer Müller
(Lehrstuhl Informatik 4)



Sommersemester 2014



Organisatorisches

Tafelübungen

Aufgaben

Aufgaben

Rechnerübungen

Bei Problemen

Entwicklungsumgebung

Hardware

Ausleihen

Bibliothek

Verzeichnisse

AVR Studio

Anleitung

CIP Login

AVR-Studio einrichten

Projekt Anlegen

Flashen

Debuggen

Abgeben



- Tafelübungen:
 - Vorstellung der neuen Aufgabe
 - Praxisnahe Vertiefung der Vorlesungsstoffs
 - ggf. Entwicklung einer Lösungsskizze
 - Besprechung der alten Aufgabe
 - Die Folien sind nicht unbedingt zum Selbststudium geeignet
 - Termine: https://www4.cs.fau.de/Lehre/SS14/V_SPIC/#woch
 - Übersicht: https://www4.cs.fau.de/Lehre/SS14/V_SPIC/#sem



- 9 Aufgaben
 - 5 x SPiCboard
 - 4 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 übersetzt nicht: 0 Punkte
 - Bei Warnungen des Compilers: Je Warnung -2 Punkt
 - 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 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
- Bonuspunkte können nicht in das nächste Semester mitgenommen werden



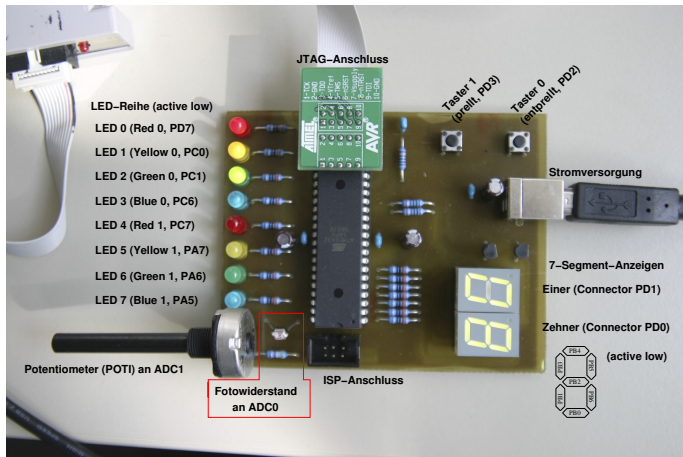
- Rechnerübungen (Raum: 01.153):
 - Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
 - In den ersten 30 Minuten (bis XX:45) haben Angemeldete Vorrang. Freie Plätze auf FCFS-Basis.
 - Falls 30 Minuten nach Beginn der Rechnerübung (also um XX:45) niemand anwesend ist, kann der Übungsleiter gehen.
 - Termine:
https://www4.cs.fau.de/Lehre/SS14/V_SPiC/#woch



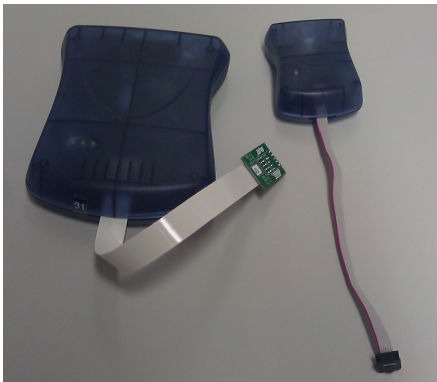
- Diese Folien konsultieren
 - Häufig gestellte Fragen (FAQ) und Antworten:
https://www4.cs.fau.de/Lehre/SS14/V_SPIC/Uebung/faq.shtml
 - Fragen zu Übungsaufgaben im EEI-Forum posten (darf auch von anderen Studienrichtungen verwendet werden!):
<https://eei.fsi.uni-erlangen.de/forum/forum/16>
 - Bei speziellen Fragen Mail an Mailingliste, die alle Übungsleiter erreicht:
i4spic@cs.fau.de
- ⇒ Zum Beispiel auch, wenn kein Übungsleiter auftaucht



- 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



- 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 Jungunst, Büro 0.046 (Erdgeschoss RRZE-Gebäude)
 - Übliche Bürozeiten: von 8:00 bis 15:00
 - <https://www4.cs.fau.de/~junguns/>
- In 01.155N befinden sich weitere Windows-Rechner



- `libspicboard`: Funktionsbibliothek zur einfachen 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/SS14/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 \\fai03\LOGINNAME
 - P:\ unter \\fai03\i4spichome
 - Q:\ unter \\fai03\i4spicpub



- Programmentwicklung unter AVR Studio 5.1 von Atmel unter Windows
- Vereint Editor, Compiler und Debugger in einer Umgebung
- Compiler: Cross-Compiler, der bei Ausführung auf Intel-PC Programme für AVR-Mikrocontroller erstellt

- 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 ~> Alle Programme ~> Atmel AVR Tools ~> AVR Studio 5.1
 - Falls Windows-Firewall einige Funktionen blockiert, auf “Abbrechen” klicken
 - Importieren der Projektvorlage (einmalig):
 - File ~> Import ~> Project Template...
 - Q:\tools\SPiC_Template5.zip
 - Add to folder: <Root>
 - OK
- ⇒ Successfully imported project template



- Pro Übungsaufgabe ein neues Projekt anlegen:
 - File ~> New ~> 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 ~> Add New Item...
 - Dateityp: C File
 - Name: siehe Aufgabenstellung, jetzt test.c (Achtung: Kleinschreibung!)
 - Add



Programmieren (1)

- Beispielprogramm, um erste grüne LED einzuschalten:

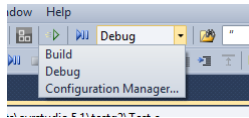
```
1 #include <led.h>
2
3 void main(void) {
4     sb_led_on(GREEN0);
5     while(1) { /* Endlosschleife */
6     }
7 }
```

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



Programmieren (2)

- **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 Werkzeugleiste

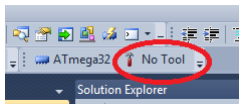


- Letztendlich soll jede Aufgabe mit Build kompiliert und getestet werden
- ⇒ *Die Build-Konfiguration wird von uns bewertet!*



Flashen mit Programmierer

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



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



Debuggen (1)

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



Debuggen (2)

- 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 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



- 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 μ C 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



Abgeben (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 \rightsquigarrow Alle Programme \rightsquigarrow PuTTY \rightsquigarrow 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

