

Übungen zu Systemnahe Programmierung in C (SPiC)

Wanja Hofer, Moritz Strübe, Rainer Müller
(Lehrstuhl Informatik 4)



Sommersemester 2012



Organisatorisches

Tafelübungen

Reine Rechnerübungen

Bonuspunkte

Bei Problemen

Hardware-Entwicklungsumgebung

Ausleihen

Software-Umgebung

Bibliothek

Verzeichnisse

AVR Studio

Programmieren

Debuggen

Flashen

Abgeben



- Zur Bearbeitung der Übungen ist ein Windows-Login nötig
- Jetzt Passwort setzen:
 - Im Raum 01.155 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`
- Passwort wird erst nach 10 Minuten aktiv, erst dann Einloggen an Windows-Rechnern möglich



Organatorisches: Tafelübungen

- Tafelübungen:
 - Vorstellung der neuen Aufgabe, ggf. gemeinsame Entwicklung einer Lösungsskizze
 - Besprechung der alten Aufgabe mit Lösungsvorstellung durch Studenten, Hinweis auf häufig gemachte Fehler
 - Keine Anwesenheitspflicht; trotzdem Anwesenheitsliste, da es bei unentschuldigter Abwesenheit bei Lösungsvorstellung ggf. 0 Punkte auf die Aufgabe gibt
 - Ebenfalls 0 Punkte bei "abgeschriebenen" Lösungen; Lösung wird automatisch auf Ähnlichkeit mit allen anderen, auch älteren Lösungen verglichen
 - Termine:
http://www4.cs.fau.de/Lehre/SS12/V_SPiC/#woch
 - Wochenrhythmus:
http://www4.cs.fau.de/Lehre/SS12/V_SPiC/#sem
- Exakte Zeit dieser Tafelübung: ab XX:00, XX:15 oder XX:30?



- Reine Rechnerübungen (Raum: 01.153 und 01.155N):
 - Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
 - Falls 30 Minuten nach Beginn der Rechnerübung (also um XX:45) niemand anwesend ist, kann der Übungsleiter gehen.
 - Termine:
http://www4.cs.fau.de/Lehre/SS12/V_SPiC/#woch



- Bonuspunkte:
 - Abgegebene Aufgaben werden bepunktet
 - Umrechnung in Bonuspunkte für die Klausur (bis zu 10%)
 - *Bestehen* der Klausur durch Bonuspunkte nicht möglich
 - Bonuspunkte gibt es ab der Hälfte der erreichbaren Bonuspunkte

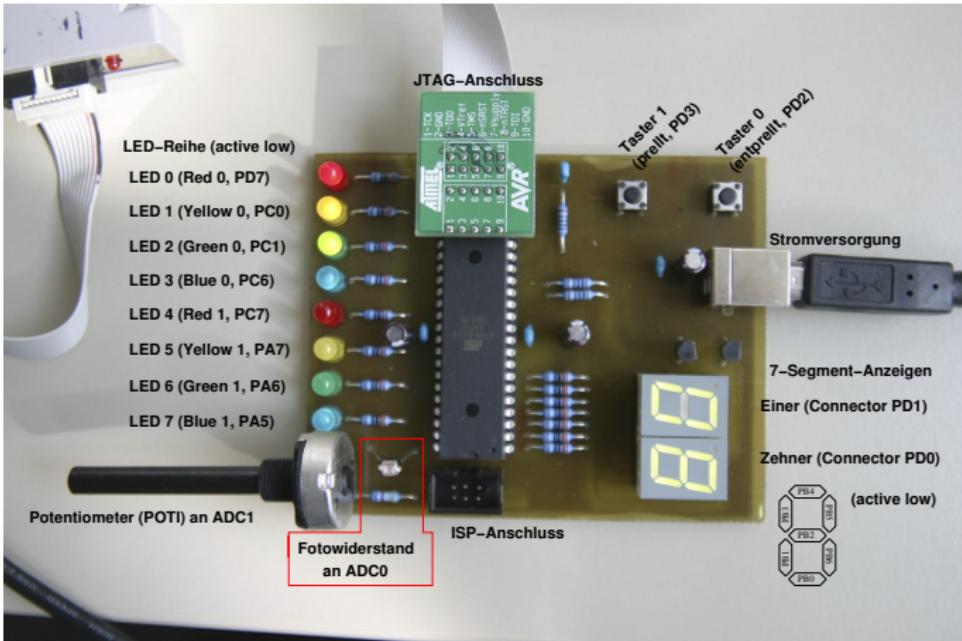


1. Diese Folien konsultieren
2. Häufig gestellte Fragen (FAQ) und Antworten (werden laufend erweitert):
http://www4.cs.fau.de/Lehre/SS12/V_SPIC/Uebung/faq.shtml
3. Fragen zu Übungsaufgaben im EEI-Forum posten; Übungsleiter lesen mit und antworten, falls Studenten nicht oder falsch antworten:
<http://eei.fsi.uni-erlangen.de/forum/forum/16>
4. Bei speziellen Fragen Mail an Mailingliste, die alle Übungsleiter erreicht:
i4spic@cs.fau.de



Hardware-Entwicklungsumgebung (1)

- Speziell für (G)SPiC angefertigte SPiCboards mit AVR-ATmega32-Mikrocontroller (hier: 1 MHz)



Hardware-Entwicklungsumgebung (2)

- 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 Tafelübungen und Rechnerübungen
- Selbständige Bearbeitung teilweise nötig
- Eigenes SPiCboard: Anfertigung am Lötabend
- Eigener Programmierer: Kauf am Lötabend oder gebraucht im Forum: <http://eei.fsi.uni-erlangen.de/forum/post/3208>
- Ausleihe von SPiCboard, Kabeln und Programmierer/Debugger für 1 Tag möglich:
 - Bei Harald Junggunst, Büro 0.046 (Erdgeschoss RRZE-Gebäude)
 - Übliche Bürozeiten: von 8:00 bis 15:00
 - <http://www4.cs.fau.de/~jungguns/>
- 01.155N hat viele freie Slots: Belegungsplan im UnivIS



- **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:
http://www4.cs.fau.de/Lehre/SS12/V_SPIC/Uebung/doc



- Projektverzeichnis pro Student:
 - Unter Windows: P:\
 - Unter Linux: /proj/i4spic/LOGINNAME/
 - Lösungen in Unterverzeichnissen **aufgabeX** entwickeln;
Abgabeprogramm sucht dort
 - Verzeichnis nur durch den Studenten lesbar
 - Erzeugung automatisch nach Waffel-Anmeldung innerhalb eines Tages
- Heimverzeichnis:
 - Unter Windows: H:\
 - Entspricht dem Heimverzeichnis ~ unter Linux



- Vorgabeverzeichnis für alle Studenten:
 - Unter Windows: Q:\
 - Unter Linux: /proj/i4spic/pub/
 - 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



- 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



- 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_Template.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**, jetzt **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 **blink.c** (Achtung: Kleinschreibung!)
 - Add



Software-Umgebung: Programmieren (1)

- Auf Mikrocontrollern ist die `main()`-Funktion normalerweise vom Typ `void main(void);`
- Sollte niemals zurückkehren (wohin?), daher kein Rückgabewert
- 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
 - ⇒ Kompilierendes Programm nur, wenn unten steht: **Build succeeded.**
 - ⇒ Fehlermeldungen erscheinen ggf. unten



- *Achtung:* Zwei verschiedene Compiler-Profile: **Build** und **Debug**
 - Unterschied: **Build** optimiert den entstehenden Binärkode, **Debug** nicht
 - Letztendlich soll jede Aufgabe mit **Build** kompiliert und getestet werden
- ⇒ *Die Build-Konfiguration wird von uns bewertet!*
- Nur zu Debug-Zwecken während der Entwicklung soll ggf. die **Debug**-Konfiguration verwendet werden
 - Beispiel: Compiler optimiert bei **Build** überflüssige Codezeile weg; Debugger kann deswegen dort nicht an einem Breakpoint anhalten
 - Umstellung des Profils in Drop-Down-Box rechts neben dem Play-Button in der Werkzeugleiste



- 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.)



- 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



- Flashen: Kompiliertes Programm in den Speicher des Mikrocontrollers kopieren
- Analog zum Debuggen
- Programmierer auswählen:
 - Project ↳ aufgabeX Properties
 - Tool ↳ Selected Debugger ↳ AVRISP mkII
 - ISP Clock: 150,00 kHz
 - File ↳ Save Selected Items
- Direkt in den Speicher kopieren und laufen lassen: Debug ↳ Continue (F5)
- (Beim ersten Mal ggf. Firmware-Upgrade durchführen lassen.)



Software-Umgebung: Binärabbild flashen

- 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
 - 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



Software-Umgebung: 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 ↵ Alle Programme ↵ PuTTY ↵ PuTTY
 - Host Name: `fau10sr0` bzw. von Zuhause `fau10sr0.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/pub/abgabe aufgabe0`
- Wichtig: Grüner Text signalisiert erfolgreiche Abgabe, roter Text einen Fehler!



- Anzeige der abgegebenen Aufgabe, dabei `aufgabe0` entsprechend ersetzen:
`/proj/i4spic/pub/showabgabe aufgabe0`
- Zeigt abgegebene Version an
- Zeigt ggf. Unterschied zwischen abgegebener Version und Version im Projektverzeichnis `P:\aufgabeX` an

