

# Übungen zu Grundlagen der Systemnahen Programmierung in C (GSPiC)

Rainer Müller  
(Lehrstuhl Informatik 4)



Sommersemester 2016



## Organisatorisches: Tafelübungen

- Tafelübungen (Raum: 01.153 oder 01.155N) im Wochenrhythmus abwechselnd:
  - Vorstellung der neuen Aufgabe, ggf. gemeinsame Entwicklung einer Lösungsskizze
  - Besprechung der alten Aufgabe mit Lösungsvorstellung durch Studierende, 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; Automatisierter Vergleich mit allen anderen (auch älteren) Lösungen
  - Termine:  
[http://www4.cs.fau.de/Lehre/SS16/V\\_GSPIC/#wochenplanung](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/#wochenplanung)
  - Wochenrhythmus:  
[http://www4.cs.fau.de/Lehre/SS16/V\\_GSPIC/#semesterplanung](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/#semesterplanung)
- Exakte Zeit dieser Tafelübung: ab XX:00, XX:15 oder XX:30?



## Prolog: Windows-Login

- 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:  
`cip-set-windows-password`
- 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 Ausschuchen mit folgendem Kommando:  
`pwgen -s 12`



## Organisatorisches: Reine Rechnerübungen

- Reine Rechnerübungen (Raum: 01.153 oder 01.155N):
  - Unterstützung durch Übungsleiter bei der Aufgabebearbeitung
  - 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/SS16/V\\_GSPIC/#wochenplanung](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/#wochenplanung)



- Bearbeitung teils einzeln, teils in Zweiergruppen (siehe Aufgabenstellung)
  - bei Teamarbeit müssen beide Partner in **derselben** Tafelübung sein
- Verschiedene Abgabezeitpunkte für jede Übungsgruppe
  - Bearbeitungszeit immer bis zum Tag vor der Herausgabe der nächsten Aufgabe
  - Abgabezeitpunkt um 18:00
  - eigener Abgabetermin kann per Skript abgefragt werden (siehe Folie 24)
  - Übersicht:  
[http://www4.cs.fau.de/Lehre/SS16/V\\_GSPIC/#semesterplanung](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/#semesterplanung)



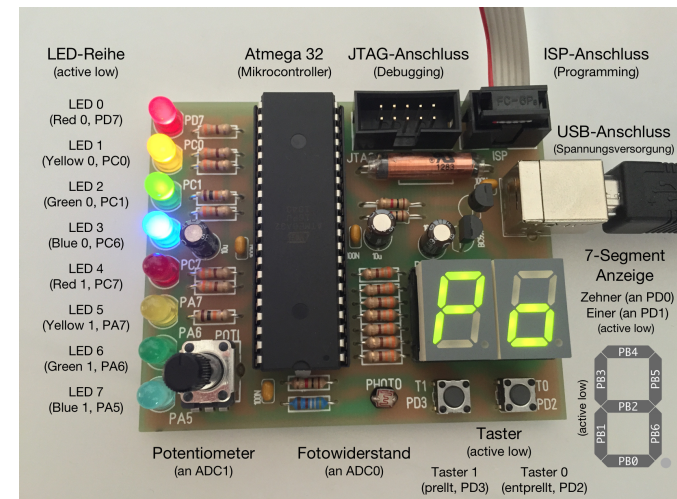
- Bonuspunkte:
  - Abgegebene Aufgaben werden bepunktet
  - Umrechnung in Bonuspunkte für die Klausur
  - *Bestehen* der Klausur nur durch Bonuspunkte nicht möglich
  - Details:  
[http://www4.cs.fau.de/Lehre/SS16/V\\_GSPIC/Pruefung/#bonus](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/Pruefung/#bonus)



1. Diese Folien konsultieren
2. Häufig gestellte Fragen (FAQ) und Antworten (werden laufend erweitert):  
[http://www4.cs.fau.de/Lehre/SS16/V\\_GSPIC/Uebung/faq.shtml](http://www4.cs.fau.de/Lehre/SS16/V_GSPIC/Uebung/faq.shtml)
3. Fragen zu Übungsaufgaben im EEI-Forum posten; Übungsleiter lesen mit und antworten, falls Studierende 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](mailto:i4spic@cs.fau.de)



- 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



## Hardware-Entwicklungsumgebung: Ausleihen

- Bearbeitung der Aufgaben größtenteils während der Tafelübungen und Rechnerübungen
- Eigenes SPiCboard: Anfertigung am Lötabend
- Eigener Programmierer: Online bestellen 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/>



## Software-Umgebung: Bibliothek

- `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/SS16/V\\_GSPiC/Uebung/doc](http://www4.cs.fau.de/Lehre/SS16/V_GSPiC/Uebung/doc)



## Software-Umgebung: Verzeichnisse (1)

- Projektverzeichnis pro Student/Studentin:
  - Unter Windows: `R:\`
  - Unter Linux: `/proj/i4gspic/LOGINNAME/`
  - Lösungen in Unterverzeichnissen `aufgabeX` entwickeln; Abgabeprogramm sucht dort
  - Verzeichnis nur für den/die jeweilige/n Student/Studentin lesbar
  - Erzeugung automatisch nach Waffel-Anmeldung innerhalb eines Tages
- Heimverzeichnis:
  - Unter Windows: `Z:\`
  - Entspricht dem Heimverzeichnis `~` unter Linux



- Vorgabeverzeichnis für alle Studierenden:
  - Unter Windows: S:\
  - Unter Linux: /proj/i4gspic/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:\, R:\, S:\ nicht angezeigt wird:
  - Windows Explorer – Computer – Map network drive
  - Z:\ unter `\\fai03\LOGINNAME`
  - R:\ unter `\\fai03\i4gspichome`
  - S:\ unter `\\fai03\i4gspicpub`



- Programmentwicklung unter 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



- Projektordner erstellen und Vorlage kopieren:
  - Ordner für aktuelle Aufgabe erstellen
    - R:\`aufgabeX\`
  - Die Datei
    - S:\`vorlage.cproj`in das Verzeichnis der aktuellen Aufgabe kopieren
    - R:\`aufgabeX\`
  - Projektdatei umbenennen entsprechend der Aufgabe
    - `vorlage.cproj` → `aufgabeX.cproj`
- Projekt in Atmel Studio öffnen:
  - Start von Atmel Studio über:
    - Start – All Programs – Atmel Studio 7.0 – Atmel Studio 7.0
  - Beim ersten Start öffnet sich ein Dialog, in dem “Standard” als User Interface ausgewählt werden sollte
  - File – Open → Project/Solution...
  - R:\`aufgabeX\aufgabeX.cproj` auswählen
  - 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



## 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:

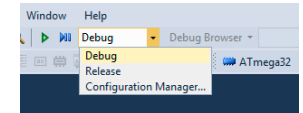
```
1 #include <led.h>
2
3 void main(void) {
4     sb_led_on(GREEN0);
5     while(1) { /* Endlosschleife */
6     }
7 }
```

- Programm kompilieren mit Build – Build Solution
- ⇒ Kompilierendes Programm nur, wenn unten steht: **Build succeeded.**
- ⇒ Fehlermeldungen erscheinen ggf. unten



## Software-Umgebung: Programmieren (2)

- *Achtung:* Zwei verschiedene Compiler-Profile: **Rele**ase und **De**bug
  - **Rele**ase optimiert den entstehenden Binärcode
  - **De**bug nicht
- Letztendlich soll jede Aufgabe mit **Rele**ase kompiliert und getestet werden
- ⇒ *Die Rele*ase-Konfiguration wird von uns bewertet!
- Nur zu Debug-Zwecken während der Entwicklung soll ggf. die **De**bug-Konfiguration verwendet werden
- Beispiel: Compiler optimiert bei **Rele**ase 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



## Software-Umgebung: 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



## Software-Umgebung: Debuggen (2)

- 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



## Software-Umgebung: Flashen mit Programmierer

- 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: Binärbild flashen

- Nötig, um vorgefertigte Binärbilder (.hex-Images) zu testen, z. B. Binärmusterlösungen unter S:\aufgabeX
- Möglich mit Debugger (ICE) oder Programmierer (ISP)
  - Tools – Device Programming
  - Tool: JTAGICE mkII bzw. AVRISP mkII
  - Device: ATmega32
  - Interface: JTAG bzw. ISP
  - Apply
  - Verbindung überprüfen mit Device ID – Read
- ⇒ Ergebnis: 0x1E9502
  - 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: faui0sr0 bzw. von Zuhause faui0sr0.cs.fau.de (alternativ: faui00[a-y] oder faui06[a-q])
  - 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/i4gspic/bin/submit aufgabe0`
- Wichtig:  
Grüner Text signalisiert erfolgreiche Abgabe, roter Text einen Fehler!



## Software-Umgebung: Abgeben (2)

- Den Quelltext der abgegebenen Aufgabe nochmal anzeigen, dabei aufgabe0 entsprechend ersetzen:  
`/proj/i4gspic/bin/show-submission aufgabe0`
- Unterschiede zwischen abgegebener Version und Version im Projektverzeichnis R:\aufgabeX anzeigen:  
`/proj/i4gspic/bin/show-submission aufgabe0 -d`
- Den eigenen Abgabetermin ermitteln:  
`/proj/i4gspic/bin/get-deadline aufgabe0`

