Übungen zu Grundlagen der Systemnahen Programmierung in C (GSPiC)

Rainer Müller
(Lehrstuhl Informatik 4)

Sommersemester 2017

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 Aussuchen mit folgendem Kommando:

```
pwgen -s 12
```

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/SS17/V_GSPIC/#wochenplanung`
  - Wochenrhythmus:
    `http://www4.cs.fau.de/Lehre/SS17/V_GSPIC/#semesterplanung`
- Exakte Zeit dieser Tafelübung: ab XX:00, XX:15 oder XX:30?

Organisatorisches: Reine Rechnerübungen

- Reine Rechnerübungen (Raum: 01.153 oder 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/SS17/V_GSPIC/#wochenplanung`

Organisatorisches: Abgaben

- 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 23)
  - Übersicht:
    `http://www4.cs.fau.de/Lehre/SS17/V_GSPIC/#semesterplanung`

Organisatorisches: Bonuspunkte

- 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/SS17/V_GSPIC/Pruefung/#bonus`

Organisatorisches: Bei Problemen

1. Diese Folien konsultieren
2. Häufig gestellte Fragen (FAQ) und Antworten (werden laufend erweitert):

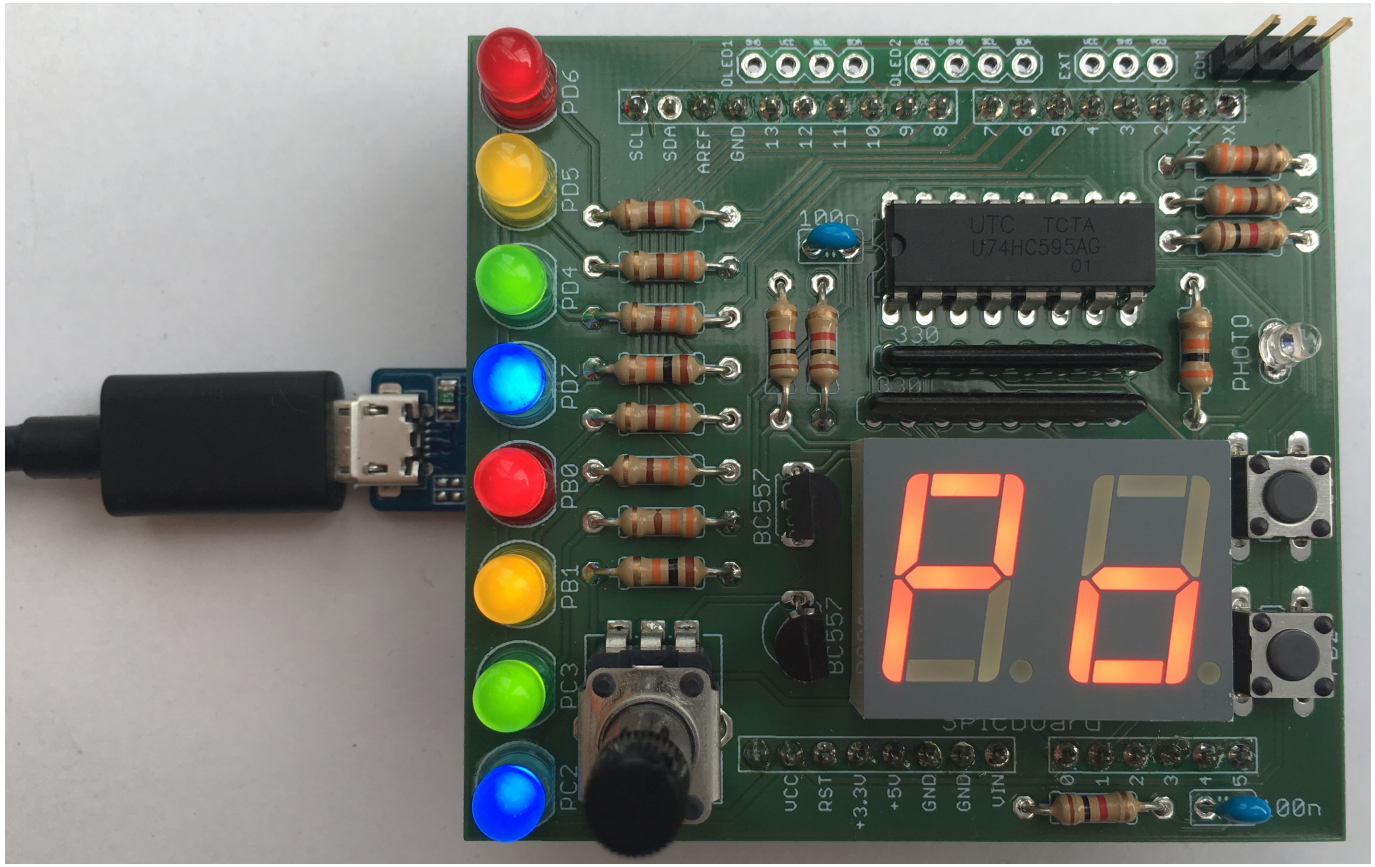
   `http://www4.cs.fau.de/Lehre/SS17/V_GSPIC/Uebung/faq/`
3. Fragen zu Übungsaufgaben im EEI-Forum posten; Übungsleiter lesen mit und antworten, falls Studierende nicht oder falsch antworten:

   `http://eei.fsi.uni-erlangen.de/forum/forum/16`
4. Bei speziellen Fragen Mail an Mailingliste, die alle Übungsleiter erreicht:

   `i4spic@cs.fau.de`

Hardware: SPiCboard

- ATmega328PB Xplained Mini:
  Mikrocontroller-Board mit integriertem Programmer/Debugger
- Speziell für (G)SPiC angefertigte SPiCboards als Erweiterungsplatine



Aufgabenbearbeitung

- Betreute Bearbeitung der Aufgaben während der Rechnerübungen

  $\Rightarrow$ Hardware wird zur Verfügung gestellt
- Selbständige Bearbeitung teilweise nötig

  $\Rightarrow$ Eigenes SPiCboard: Anfertigung am Lötabend

  $\Rightarrow$ SPiCboard Simulator: SPiCsim

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/SS17/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

Software-Umgebung: Verzeichnisse (2)

- 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 `\\faui03\LOGINNAME`
  - `R:\` unter `\\faui03\i4gspichome`
  - `S:\` unter `\\faui03\i4gspicpub`

Software-Umgebung: Atmel Studio

- 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

Software-Umgebung: Projekt anlegen

- 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

Software-Umgebung: C-Dateien hinzufügen

- 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.`

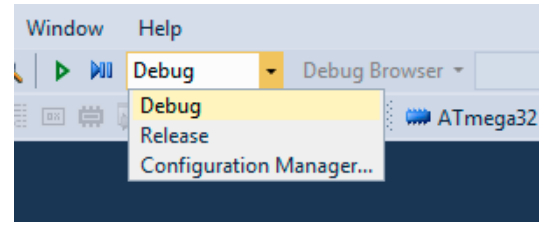
⇒ Fehlermeldungen erscheinen ggf. unten

Software-Umgebung: Programmieren (2)

- *Achtung:* Zwei verschiedene Compiler-Profile: `Release` und `Debug`
  - `Release` optimiert den entstehenden Binärcode
  - `Debug` nicht
- Letztendlich soll jede Aufgabe mit `Release` kompiliert und getestet werden

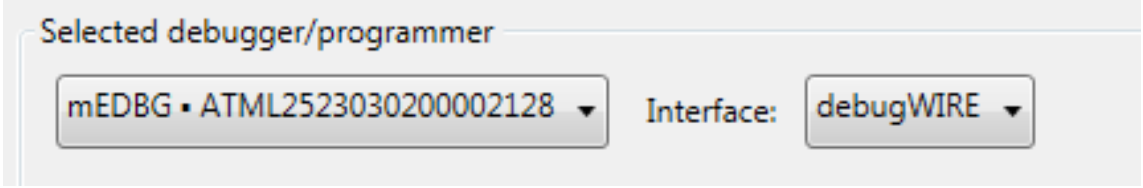
⇒ *Die `Release`-Konfiguration wird von uns bewertet!*

- Nur zu Debug-Zwecken während der Entwicklung soll ggf. die `Debug`-Konfiguration verwendet werden
- Beispiel: Compiler optimiert bei `Release` 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: Verbindung Einrichten

- On-Chip-Debugger zum Untersuchen des Programmablaufs “live” auf dem Board
- Verbindung per debugWIRE auswählen:
  - Project – aufgabeX Properties
  - Tool – Selected Debugger
    - ⇝ mEDBG
    - ⇝ debugWIRE



Software-Umgebung: Flashen und Ausführen

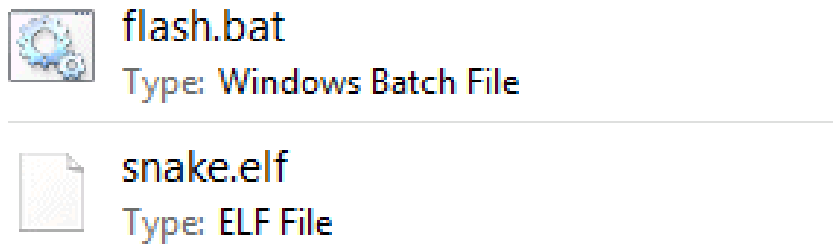
- Flashen: Kompiliertes Programm in den Speicher des Mikrocontrollers kopieren
- Direkt in den Speicher kopieren und laufen lassen:
  Debug – Start without Debugging
- Nach erfolgreichem Flashen führt das Board das Programm direkt aus
- Ein Neustart des Programms ist durch Trennen und Wiederherstellen der USB-Stromversorgung möglich

Software-Umgebung: Debuggen

- 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 bestimmen 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: Binärabbild flashen

- Nötig, um vorgefertigte Binärabbilder (`.elf`-Images) zu testen, z. B. Binärmusterlösungen unter `S:\aufgabeX`
- Möglich mit Skript `flash.bat` im jeweiligen Verzeichnis, Ausführen mit Doppelklick



- Nach erfolgreichem Flashen führt das Board das Programm direkt aus
- Ein Neustart des Programms ist durch Trennen und Wiederherstellen 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 `aufgabeX` entsprechend ersetzen:

  ```
  /proj/i4gspic/bin/submit aufgabeX
  ```

- Wichtig:
  Grüner Text signalisiert erfolgreiche Abgabe, roter Text einen Fehler!

Software-Umgebung: Abgeben (2)

- Den Quelltext der abgegebenen Aufgabe nochmal anzeigen, dabei `aufgabeX` entsprechend ersetzen:

  ```
  /proj/i4gspic/bin/show-submission aufgabeX
  ```

- Unterschiede zwischen abgegebener Version und Version im Projektverzeichnis `R:\aufgabeX` anzeigen:

  ```
  /proj/i4gspic/bin/show-submission aufgabeX -d
  ```

- Den eigenen Abgabetermin ermitteln:

  ```
  /proj/i4gspic/bin/get-deadline aufgabeX
  ```