Übungen zu Systemnahe Programmierung in C (SPiC)

Moritz Strübe, Rainer Müller
(Lehrstuhl Informatik 4)

Wintersemester 2013/2014

Inhalt

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:
    `/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
```

Organisatorisches: Tafelübungen

- Tafelübungen:
  - Besprechung der alten Aufgabe, Hinweis auf häufig gemachte Fehler
  - Keine Anwesenheitspflicht; bei unentschuldigter Abwesenheit ggf. 0 Punkte auf die Aufgabe
  - Ebenfalls 0 Punkte bei "abgeschriebenen" Lösungen; Achtung: Abgabe von Lösungen aus dem letzten Semester ist heikel.
  - Vorstellung der neuen Aufgabe, ggf. gemeinsame Entwicklung einer Lösungsskizze
  - Termine: `http://www4.cs.fau.de/Lehre/WS13/V_SPIC/#woch`
  - Wochenrhythmus: `http://www4.cs.fau.de/Lehre/WS13/V_SPIC/#sem`

Organisatorisches: Reine Rechnerübungen

- Reine Rechnerübung Mi 18:00 – 19:30:
  - Unterstützung durch Übungsleiter bei der Aufgabenbearbeitung
  - Falls 30 Minuten nach Beginn der Rechnerübung (also um 18:30) niemand anwesend ist, kann der Übungsleiter gehen.

Organisatorisches: Bonuspunkte

- 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

Organisatorisches: Bei Problemen

- Diese Folien konsultieren

  $\Rightarrow$ Die Folien sind kein Skript!
- Häufig gestellte Fragen (FAQ) und Antworten:

  `http://www4.cs.fau.de/Lehre/WS13/V_SPIC/Uebung/faq.shtml`
- Fragen zu Übungsaufgaben im EEI-Forum posten; Übungsleiter lesen mit:

  `http://eei.fsi.uni-erlangen.de/forum/forum/16`
- Bei speziellen Fragen Mail an Mailingliste:

  i4spic@cs.fau.de

Hardware-Entwicklungsumgebung / Aufgaben

- 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
  - `http://www4.cs.fau.de/~jungguns/`

Software-Umgebung: Bibliothek

- `libspicboard`: Funktionsbibliothek zur einfachen Ansteuerung der Hardware
- Dokumentation online:

  `http://www4.cs.fau.de/Lehre/WS13/V_SPIC/Uebung/doc`

Software-Umgebung: Verzeichnisse (1)

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

Software-Umgebung: Verzeichnisse (2)

- Vorgabeverzeichnis für alle Studenten:
  - Unter Linux: `/proj/i4spic/pub/`
  - Unter 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 `\\faui03\LOGINNAME`
  - `P:\` unter `\\faui03\i4spichome`
  - `Q:\` unter `\\faui03\i4spicpub`

Software-Umgebung: AVR-Studio-Einrichtung

- Start von AVR Studio über: Start $\rightsquigarrow$ Alle Programme $\rightsquigarrow$ Atmel AVR Tools $\rightsquigarrow$ AVR Studio 5.1
- Falls Windows-Firewall einige Funktionen blockiert, auf "Abbrechen" klicken
- Importieren der Projektvorlage (einmalig):
  - File $\rightsquigarrow$ Import $\rightsquigarrow$ Project Template...
  - `Q:\tools\SPiC_Template.zip`
  - Add to folder: `<Root>`
  - OK

$\Rightarrow$ Successfully imported project template

Software-Umgebung: AVR-Studio-Projekt

- Pro Übungsaufgabe ein neues Projekt anlegen:
  - File ⇝ New ⇝ Project...
  - Projekttyp: `(G)SPiC-Projekt`
  - Name: `aufgabeX`, jetzt `aufgabe1` (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 `zaehler.c` (Achtung: Kleinschreibung!)
  - Add

Software-Umgebung: Programmieren

- *Achtung:* Zwei verschiedene Compiler-Profile: `Build` und `Debug`
- Unterschied: `Build` optimiert den Binärcode, `Debug` nicht

⇒ *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

Software-Umgebung: 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)

Software-Umgebung: 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 bestimmen 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

Software-Umgebung: Flashen mit Programmierer

- Flashen: Kompiliertes Programm in den Speicher des Mikrocontrollers kopieren
- Analog zum Debuggen
- Programmierer auswählen:
  - Project $\rightsquigarrow$ aufgabeX Properties
  - Tool $\rightsquigarrow$ Selected Debugger $\rightsquigarrow$ AVRISP mkII
  - ISP Clock: 150,00 kHz
  - File $\rightsquigarrow$ Save Selected Items
- Direkt in den Speicher kopieren und laufen lassen: Debug $\rightsquigarrow$ Continue (F5)

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 $\rightsquigarrow$ AVR Programming
  - Tool: JTAGICE mkII bzw. AVRISP mkII
  - Device: ATmega32
  - Interface: JTAG bzw. ISP
  - Apply
  - Verbindung überprüfen mit Device ID – Read

  $\rightsquigarrow$ Ergebnis: `0x1E 0x95 0x02`

  $\Rightarrow$ Eignet sich gut um schnell 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-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 $\rightsquigarrow$ 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 aufgabe1
  ```

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

Software-Umgebung: Abgeben (2)

- Anzeige der abgegebenen Aufgabe, dabei `aufgabe1` entsprechend ersetzen:

  ```
  /proj/i4spic/bin/show-submission aufgabe1
  ```

- Zeigt abgegebene Version an
- Zeigt ggf. Unterschied zwischen abgegebener Version und Version im Projektverzeichnis `P:\aufgabeX` an

Aufgabe 1 - Zähler

- Tastendruck von Taster 1 zählen (Achtung nicht entprellt)
- Beim druck von Taster 0 Anzeige aktualisieren
  - 7-Segmentanzeige und Leds

Konfiguration der Pins

- Jeder I/O-Port des AVR-µC wird durch drei 8-bit Register gesteuert:
  - Datenrichtungsregister (`DDRx` = data direction register)
  - Datenregister (`PORTx` = port output register)
  - Port Eingabe Register (`PINx` = port input register, nur-lesbar)
- Jedem Anschluss-Pin ist ein Bit in jedem der 3 Register zugeordnet

I/O-Port-Register

- `DDRx`: hier konfiguriert man einen Pin i von Port x als Ein- oder Ausgang
  - `Bit i = 1` → Pin i als Ausgang verwenden
  - `Bit i = 0` → Pin i als Eingang verwenden
- `PORTx`: Auswirkung abhängig von `DDRx`:
  - ist Pin i als Ausgang konfiguriert, so steuert Bit i im `PORTx` Register ob am Pin i ein high- oder ein low-Pegel erzeugt werden soll
    - `Bit i = 1` → high-Pegel an Pin i
    - `Bit i = 0` → low-Pegel an Pin i
  - ist Pin i als Eingang konfiguriert, so kann man einen internen pull-up-Widerstand aktivieren
    - `Bit i = 1` → pull-up-Widerstand an Pin i (Pegel wird auf high gezogen)
    - `Bit i = 0` → Pin i als tri-state konfiguriert
- `PINx`: Bit i gibt den aktuellen Wert des Pin i von Port x an (nur lesbar)

Beispiel: Initialisierung eines Ports

- Pin 3 von Port B (PB3) als Ausgang konfigurieren und auf Vcc schalten:

```
1 DDRB |= (1 << 3);/* =0x08; PB3 als Ausgang nutzen... */
2 PORTB |= (1 << 3);/* ...und auf 1 (=high) setzen */
```

- Pin 2 von Port D (PD2) als Eingang nutzen, pull-up-Widerstand aktivieren und prüfen ob ein low-Pegel anliegt:

```
1 DDRD &= ~(1 << 2);/* PD2 als Eingang nutzen... */
2 PORTD|= (1 << 2);/* pull-up-Widerstand aktivieren*/
3 if ( (PIND & (1 << 2)) == 0) { /* den Zustand auslesen */
4         /* ein low Pegel liegt an, der Taster ist gedrückt */
5 }
```

- Die Initialisierung der Hardware wird in der Regel einmalig zum Programmstart durchgeführt