Übungen zu Systemnahe Programmierung in C

Abschnitt 3.3: Aufgabe (snake)

11.05.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

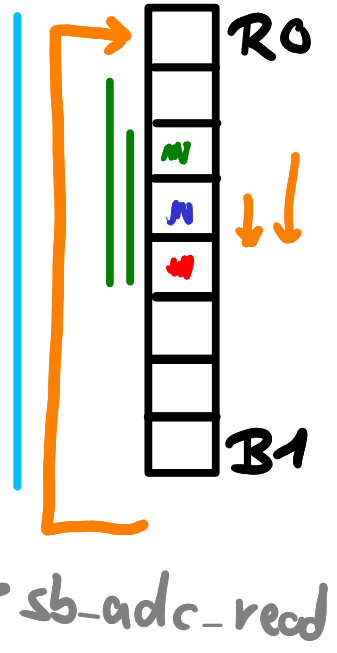
Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Aufgabe: snake



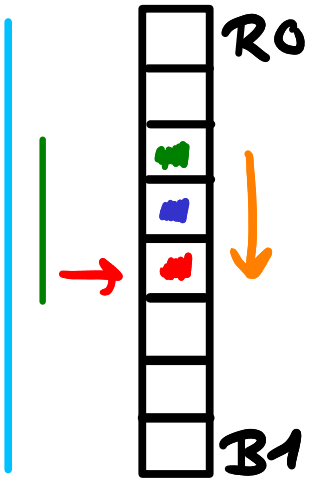
- Schlange bestehend aus benachbarten LEDs
- Länge 1 bis 5 LEDs, regelbar mit Potentiometer (`POTI`)
- Geschwindigkeit abhängig von der Umgebungshelligkeit (`PHOTO`)
  - ⇝ Je heller die Umgebung, desto schneller
- Modus der Schlange mit Taster (`BUTTON0`) umschaltbar
  - Normal: Leuchtende LEDs repräsentieren Schlange
  - Invertiert: Inaktive LEDs repräsentieren Schlange

Allgemeine Hinweise

- Variablen in Funktionen verhalten sich weitgehend wie in Java
  - ⇝ Zur Lösung der Aufgabe sind lokale Variablen ausreichend ↷3.1
- Der C-Compiler liest Dateien von oben nach unten
  - ⇝ Legen Sie die Funktionen in der folgenden Reihenfolge an:
    1. `wait()`
    2. `drawsnake()`
    3. `main()`

⇒ Details zum Kompilieren werden in der Vorlesung besprochen.

Beschreibung der Schlange



- Position des Kopfes
  - Nummer einer LED
  - Wertebereich $\{0, 1, \ldots, 7\}$
- Länge der Schlange
  - Ganzzahl aus $\{1, 2, \ldots, 5\}$
- Modus der Schlange
  - Hell oder dunkel
  - Beispielsweise durch 0 und 1 repräsentiert
- Geschwindigkeit der Schlange
  - Hier: Durchlaufzahl der Warteschleife

Zerlegung in Teilprobleme

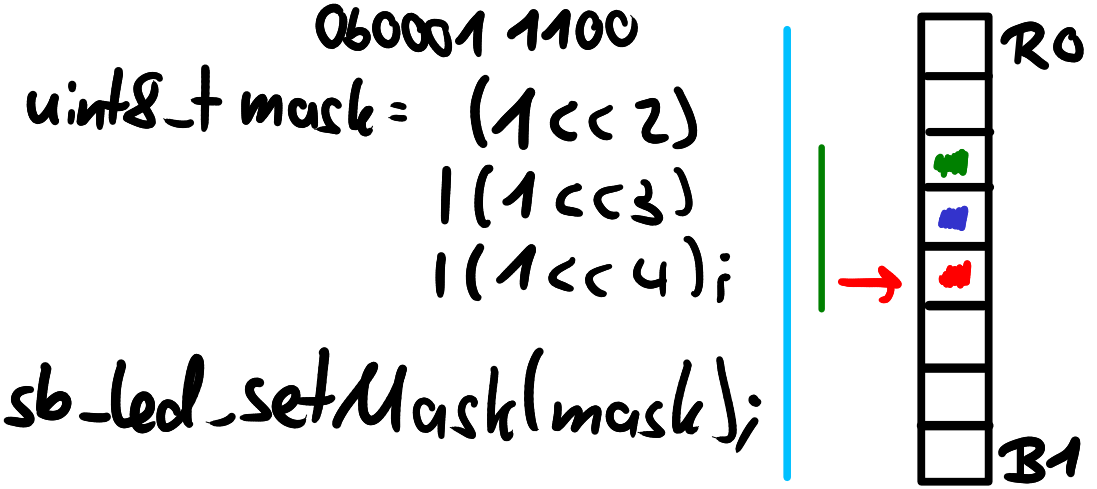
- Basisablauf: Welche Schritte wiederholen sich immer wieder?
- Vermeidung von Codeduplikation:
  - ⇝ Wiederkehrende Teilprobleme in eigene Funktionen auslagern
- Kapselung: Sichtbarkeit möglichst weit einschränken
  - Ist der Zustand nur für eine Funktion relevant?
    - ⇝ Lokale Variable
  - Greifen mehrere Funktionen auf den gleichen Zustand zu?
    - ⇝ Modullokale/globale Variable

Basisablauf snake

- Basisablauf: Schlange darstellen, Schlange bewegen, ...
- Pseudocode:

```
01 void main(void) {
02   while(1) {
03     // Berechne Laenge
04     laenge = ...
05
06     // Zeichne Schlange
07     drawSnake(kopf, laenge, modus);
08
09     // Setze Schlangenkopf weiter
10     ...
11
12     // Warte und bestimme Modus
13     ...
14
15   } // Ende der Hauptschleife
16 }
```

Darstellung der Schlange



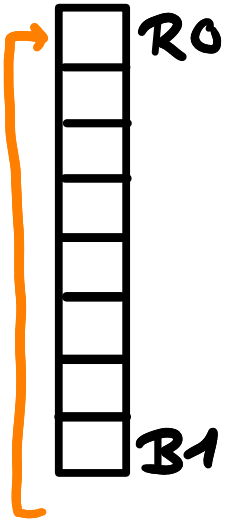
- Darstellungsparameter
  - Kopfposition
  - Länge
  - Modus
- Funktionssignatur:

```
static void drawSnake(uint8_t head, uint8_t length,
↪ uint8_t modus)
```

- Anzeige der Schlange abhängig von den Parametern
  - Normaler Modus (Helle Schlange):
    - Aktivieren der zur Schlange gehörenden LEDs
    - Deaktivieren der restlichen LEDs
  - Invertierter Modus (Dunkle Schlange): ~mask
    - Deaktivieren der zur Schlange gehörenden LEDs
    - Aktivieren der restlichen LEDs

→ 3.2, sb_led_setMask

Der Modulo Operator



- Bewegen der Schlange
  - Kopfposition abhängig von der Bewegungsrichtung anpassen
  - Problem: Was passiert am Ende der LED-Leiste?
- Eine Lösung: Der Modulooperator %
  - Divisionsrest einer Ganzzahldivision
  - **Achtung**: In C ist das Ergebnis im negativen Bereich auch negativ
  - Beispiel: `b = a % 4;`

a	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
b	-1	0	-3	-2	-1	0	1	2	3	0	1	2

Flankendetektion ohne Interrupts

- Aktives Warten zwischen Schlangenbewegungen
  - Erkennen ob der Button gedrückt wurde
  - Detektion der Flanke durch **zyklisches Abfragen** (engl. Polling) des Pegels
  - Später: Realisierung durch Interrupts