

Übungen zu Systemnahe Programmierung in C (SPiC)

Sebastian Maier
(Lehrstuhl Informatik 4)

Übung 3



Sommersemester 2016



Bits & Bytes

Bitoperationen

Shiftoperationen

Aufgabe 3: Geschicklichkeitsspiel

Aufgabenstellung & Hinweise

Flankendetektion ohne Interrupts

Hands-on: Binärzähler



Bits & Bytes

Bitoperationen

Shiftoperationen

Aufgabe 3: Geschicklichkeitsspiel

Hands-on: Binärzähler



Bitoperationen

- Übersicht:

| & | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| \wedge | 0 | 1 |
|----------|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| \sim | |
|--------|---|
| 0 | 1 |
| 1 | 0 |

- Beispiel:

| | | | |
|--------|------|------|----------|
| | 1100 | 1100 | 1100 |
| \sim | & | | \wedge |
| 1001 | 1001 | 1001 | 1001 |
| 0110 | 1000 | 1101 | 0101 |



Shiftoperationen

- Beispiel:

```
uint8_t x = 0x9d;
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

```
x <= 2;
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|

```
x >= 2;
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- Setzen von Bits:

```
(1 << 0)
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

```
(1 << 3)
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

```
(1 << 3) | (1 << 0)
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

- Achtung:

Bei signed-Variablen ist das Verhalten des `>>`-Operators nicht 100% definiert. Im Normalfall(!) werden bei negativen Werten 1er geshiftet.



Bits & Bytes

Aufgabe 3: Geschicklichkeitsspiel

Aufgabenstellung & Hinweise

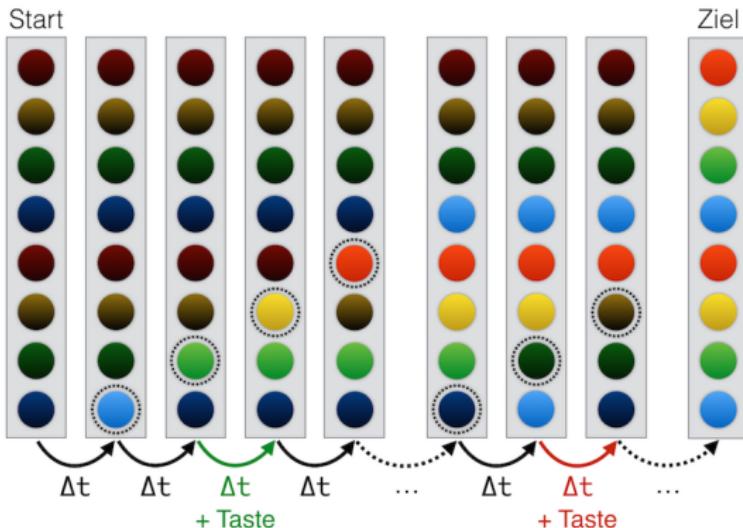
Flankendetektion ohne Interrupts

Hands-on: Binärzähler



Aufgabe 3: Geschicklichkeitsspiel (1)

- Spielcursor wandert dabei über LED-Reihe hin und her und invertiert (engl. toggle) den LED-Zustand
- LED-Zustand bleibt durch Drücken des Tasters erhalten
- Ziel: alle LEDs zum Leuchten bringen



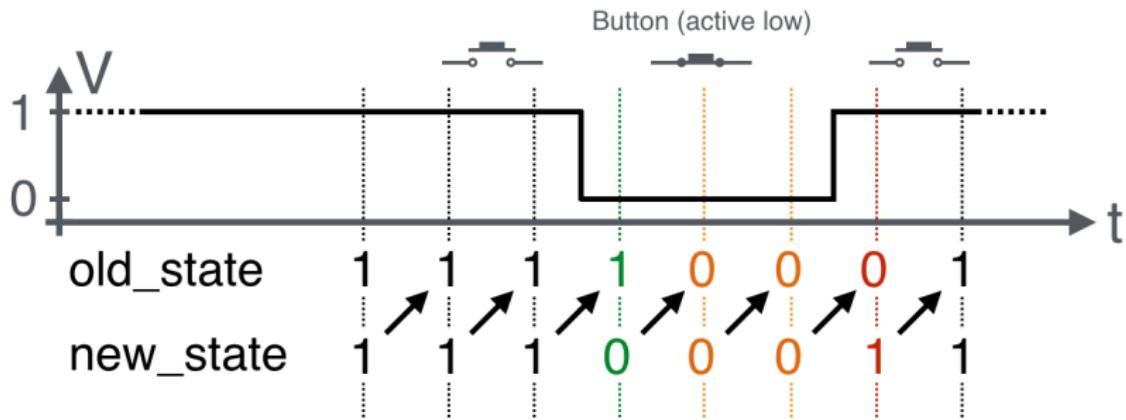
Aufgabe 3: Geschicklichkeitsspiel (2)

- Schwierigkeit (Geschwindigkeit) steigt mit jedem Level
- Nach einem Level wird eine Siegessequenz auf den LEDs dargestellt

```
1 static void play(uint8_t level)
2 static void show_win(void);
3
4 void main(void) {
5     uint8_t level = 1;
6
7     while(1){
8
9         play(level);
10
11         show_win();
12
13         // Level aktualisieren
14     }
15
16 }
```



Aufgabe 3: Flankendetektion ohne Interrupts



- Detektion der Flanke durch aktives, **zyklisches Abfragen** (engl. Polling) eines Pegels
- Unterscheidung zwischen **active-high** & **active-low** notwendig
- Später: Realisierung durch Interrupts



Bits & Bytes

Aufgabe 3: Geschicklichkeitsspiel

Hands-on: Binärzähler



Hands-on: Binärzähler

- Ausgabe des aktuellen Zählerstandes
 - binär auf LEDs 0 - 7
 - hexadezimal auf 7-Segment Anzeigen
- Zählvariable durch Tastendruck inkrementierbar
 - Button 0 inkrementiert erste Hexadezimalstelle
 - Button 1 inkrementiert zweite Hexadezimalstelle
- Betätigung der Tasten soll durch Flankendetektion erkannt werden
- *Optional:* eigene `set_all_leds()` mit invertierter LED-Reihenfolge

