# GSPiC-Aufgabe #4: Geschicklichkeitsspiel

## (12 Punkte, keine Gruppen)

Programmieren Sie ein Geschicklichkeitsspiel (Datei `gesch.c`) zum Training der Auge-Hand-Koordination. Ein Spielcursor wandert dabei über die LED-Reihe des SPiCboards. Wird der Taster 0 gedrückt, wird die LED auf die der Cursor im Moment des Tastendrucks zeigt, abhängig von ihrem vorherigen Zustand, an- bzw. ausgeschaltet. Eine bereits leuchtende LED wird wieder ausgeschaltet, eine nicht leuchtende LED wird angeschaltet. Ziel des Spiels ist es, dass alle 8 LEDs leuchten.

1. Zu Beginn des Spiels sind LED0 – LED7 ausgeschaltet.
2. Der aktuell erreichte Level wird, beginnend mit 1, auf der Sieben-Segmentanzeige dargestellt.
3. Der Spielcursor "wandert" fortlaufend von LED0 zu LED7 und wieder zurück zu LED0. Dazu wird an der aktuellen Cursorposition der Zustand der LED kurzzeitig invertiert (eine *aus*geschaltete LED wird *ein*geschaltet; eine *ein*geschaltete LED wird *aus*geschaltet).
4. Drücken von Taster 0 hält diese Invertierung fest, dass heißt die kurzzeitige Invertierung bleibt dauerhaft bestehen, auch wenn der Cursor weiter wandert.
5. Wenn alle 8 LEDs leuchten, ist das Spiel gewonnen. Es folgt die Siegessequenz, deren Ablauf durch kurzes Warten zwischen den Schritten erkennbar sein muss:
   (a) LED7 – LED0 werden hintereinander ausgeschaltet.
   (b) Der Cursor wandert einmal von LED7 zu LED0 und wieder zurück.
   (c) Die LEDs *füllen* sich von außen nach innen und werden dort beginnend wieder ausgeschaltet.
6. Das Spiel geht in den nächsten Level (die Cursorgeschwindigkeit nimmt dabei zu und nähert sich einer Maximalgeschwindigkeit) und beginnt erneut.

Ihr Programm soll in zwei Hauptteile unterteilt werden, die geeignet aus `main()` aufzurufen sind: `play()` (das eigentliche Spiel) und `show_win()` (Siegessequenz).

## Hinweise

- Nutzen Sie die `libspicboard` sowohl zur Ansteuerung von Siebensegmentanzeige und LEDs als auch zum Warten (Funktion `sb_timer_delay()` aus dem `Timer`-Modul).
- Verwenden Sie *ausschließlich* die Funktion `sb_led_setMask()`, um die LEDs anzusteuern!
- Verwenden Sie Schleifen und Bitoperationen, um die Muster für die Siegessequenz zu erstellen.
- Die Verwendung der `Button`-Funktionen der `libspicboard` ist **nicht** zulässig!
  - Konfigurieren Sie direkt die Interruptbehandlung für `BUTTON0`. Dieser ist an Pin `PD2` und damit an der externen Interruptquellen `INT0` des ATmega-Mikrocontrollers angeschlossen.
  - Ein für das Spiel relevanter Tastendruck wird durch eine fallende Flanke signalisiert.
  - Mehrfaches Drücken während der selben Cursorposition muss nicht berücksichtigt werden.
  - Die Unterbrechungsbehandlungsroutine soll möglichst kurz sein.
- Im Verzeichnis `/proj/i4gspic/pub/aufgabe4/` unter Linux bzw. in `S:\aufgabe4\` unter Windows befindet sich die Datei `gesch.elf`, welche eine Beispielimplementierung enthält.
- Ihr Programm muss mit der `Release`-Compiler-Konfiguration kompilieren und funktionieren; diese Konfiguration wird zur Bewertung herangezogen.

## Abgabezeitpunkt

| | | |
|---|---|---|
| T01 | 18.06.2018 | 18:00:00 |
| T02 | 18.06.2018 | 18:00:00 |
| T03 | 18.06.2018 | 18:00:00 |
| T04 | 19.06.2018 | 18:00:00 |
| T05 | 19.06.2018 | 18:00:00 |
| T06 | 19.06.2018 | 18:00:00 |
| T07 | 21.06.2018 | 18:00:00 |
| T08 | 21.06.2018 | 18:00:00 |