
SPiC-Aufgabe #3: 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 und kann durch “rechtzeitiges” Drücken des Tasters 0 “festgehalten” werden, so dass schließlich alle 8 LEDs leuchten und das Spiel gewonnen ist. Aber Vorsicht: Bereits eingeschaltete LEDs werden durch Tastendruck wieder ausgeschaltet! Im Detail soll sich der Spielablauf wie folgt darstellen:

1. Zu Beginn des Spiels sind LED0 – LED7 ausgeschaltet.
2. 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 *ausgeschaltete* LED wird *eingeschaltet*; eine *eingeschaltete* LED wird *ausgeschaltet*).
3. Drücken von Taster 0 hält diese Invertierung “fest”. Sie bleibt bestehen, wenn der Cursor weiter wandert.
4. Wenn alle 8 LEDs leuchten, ist das Spiel gewonnen. Es folgt die Siegessequenz, deren Schritte durch kurzes Warten dazwischen sichtbar sein müssen:
 - (a) LED0 – LED7 werden hintereinander ausgeschaltet.
 - (b) Der Cursor wandert einmal von LED0 zu LED7 und wieder zurück.
 - (c) Die LEDs “füllen” sich von LED0 zu LED7 und werden dort beginnend wieder ausgeschaltet.
 - (d) Die LEDs “füllen” sich noch einmal. Diesmal “leeren” sie sich mit LED0 beginnend.
5. Das Spiel geht in den nächsten Level (die Cursorgeschwindigkeit nimmt linear zu¹) und beginnt erneut.
6. Der aktuell erreichte Level wird, beginnend mit 1, auf der Sieben-Segmentanzeige dargestellt.

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

Beachten Sie bei der Programmierung von `play()`:

- Schreiben Sie eine Funktion `wait_key()`, welche abhängig vom Level eine bestimmte Zeit wartet und einen in dieser Zeit erfolgten Tastendruck (logisch “steigende” Flanke, d. h. ein Wechsel von `BTNRELEASED` zu `BTNPRESSED`) von `BUTTON0` geeignet zurückgibt. Beachten Sie hierbei, dass die Wartezeit nicht von der Dauer des Tastendrucks abhängen darf.
- Schreiben Sie zur Darstellung des Cursors eine Funktion `show_cursor()`, welche als Parameter den Zustand von LED0 – LED7 sowie die Cursorposition übergeben bekommt.

Hinweise

- Verwenden Sie Schleifen und Bitoperationen, um die Muster zu erstellen.
- Verwenden Sie *ausschließlich* die Funktion `sb_led_set_all_leds()`, um die LEDs anzusteuern!
- Verwenden Sie ausschließlich lokale Variablen.
- Die `libspicboard` verwendet für die Sieben-Segmentanzeige Interrupts, welche jedoch erst später besprochen werden. Binden Sie die Headerdatei `avr/interrupt.h` ein und rufen Sie am Anfang der `main()`-Funktion den Befehl `sei()` auf, um Interrupts zu aktivieren.
- Im Verzeichnis `/proj/i4spic/pub/aufgabe3/` unter Linux bzw. in `Q:\aufgabe3\` unter Windows befindet sich die Datei `gesch.hex`, welche eine Beispielimplementierung enthält.
- Ihr Programm muss mit der **Release**-Compiler-Konfiguration kompilieren und funktionieren; diese Konfiguration wird zur Bewertung herangezogen.

¹Alternativ kann die Geschwindigkeit verdoppelt werden.

Abgabezeitpunkt

T01	08.05.2016	18:00:00
T02	09.05.2016	18:00:00
T03	09.05.2016	18:00:00
T04	09.05.2016	18:00:00
T05	09.05.2016	18:00:00
T06	09.05.2016	18:00:00
T07	10.05.2016	18:00:00
T08	10.05.2016	18:00:00
T09	10.05.2016	18:00:00