GSPiC-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 *aus*geschaltete LED wird *ein*geschaltet; eine *ein*geschaltete LED wird *aus*geschaltet).
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) 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.

   Danach beginnt das Spiel von vorn.

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 einen in dieser Zeit erfolgten Tastendruck (logisch "steigende" Flanke, d. h. ein Wechsel von `BUTTONSTATE_RELEASED` zu `BUTTONSTATE_PRESSED`) 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_setAll()`, um die LEDs anzusteuern!
- Verwenden Sie *ausschließlich* lokale Variablen.
- Im Verzeichnis `/proj/i4gspic/pub/aufgabe3/` unter Linux bzw. in `S:\aufgabe3\` 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	Sonntag,	04.06.2017	18:00
T02, T03, T04	Montag,	05.06.2017	18:00
T05, T06	Dienstag,	06.06.2017	18:00
T07, T08	Donnerstag,	08.06.2017	18:00

Optional: zusätzliche Erweiterungen ohne Bewertung

Nachfolgend finden Sie Vorschläge, wie Sie das Spiel noch weiter ausbauen können. Diese dürfen mit abgegeben werden, gehen aber nicht in die Bewertung mit ein.

1. Nach der Siegessequenz geht das Spiel in den nächsten Level: die Cursorgeschwindigkeit nimmt linear zu oder alternativ kann die Geschwindigkeit auch verdoppelt werden.
2. Der aktuell erreichte Level wird, beginnend mit 1, auf der Sieben-Segmentanzeige dargestellt.
   - Die Sieben-Segmentanzeige wird mit dem Befehl `sb_7seg_showNumber()` angesteuert, siehe Dokumentation der libspicboard.
   - 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.