SPiC-Aufgabe #3: `Geschicklichkeitsspiel`

(12 Punkte, Abgabe bis Mittwoch, 02.06.2010, um 18:00, keine Gruppen)

Programmieren Sie ein Geschicklichkeitsspiel (Datei `gesch.c`) zum Training der Auge-Hand-Koordination. Ein Spielcursor wandert dabei über die LED-Reihe des SPiC-Boards 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:
   (a) LED0 – LED7 werden 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 wird verdoppelt) 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 jeder Tastendruck erkannt werden muss und 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.

Allgemeine 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/` befindet sich die Datei `gesch.hex`, welche eine Beispielimplementierung enthält.