
GSPiC-Aufgabe #3: Geschicklichkeitsspiel

(12 Punkte, Abgabe bis Freitag, 15.06.2012, um 14:00, keine Gruppen)

Programmieren Sie ein Geschicklichkeitsspiel (Datei `gesch.c`) zum Training der Auge-Hand-Koordination. Ein Spielcursor wandert dabei über die LED-Reihe des 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.

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 `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.

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.
- Im Verzeichnis `/proj/i4gspic/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 Build-Compiler-Konfiguration kompilieren und funktionieren; diese Konfiguration wird zur Bewertung herangezogen.