SPiC-Aufgabe #2: izaehler

(12 Punkte, keine Gruppen)

Schreiben Sie das Programm `izaehler.c` (Interrupt Zähler), das die Tastendrücke an Taster 0 zählt (fallende Flanke). Wird der Taster 1 gedrückt, sollen die gezählten Tastendrücke angezeigt werden. Die Einer- und Zehnerstelle wird auf der Siebensegmentanzeige und die Hunderterstelle mit den LEDs angezeigt (`LED0 = 100`, `LED1 = 200`, etc.).

- Durch Tastendruck auf Taster 0 soll der Zähler um eins inkrementiert werden.
- Die Anzeige soll bei einem Tastendruck auf Taster 1 aktiviert werden. Ändert sich der Zählerstand während die Anzeige aktiv ist, dann soll die Anzeige aktualisiert werden.
- Um Energie zu sparen, soll der Mikrocontroller so oft wie möglich in den Ruhezustand versetzt werden. Dazu soll insbesondere die Erkennung von Tastendrücken nicht durch aktives Warten, sondern durch Interrupts implementiert werden.
- Des Weiteren soll die Anzeige nach einer bestimmten Zeit ohne Tastendruck auf Taster 1 abgeschaltet werden. Die Anzeigedauer soll über das Potentiometer im Bereich von 1 bis 10 Sekunden eingestellt werden können. Auf das Abschalten der Anzeige soll ebenfalls nicht aktiv gewartet werden. Stattdessen soll ein Alarm aufgezogen werden, der dann auslöst, wenn die Anzeige abgeschalten werden soll.
- Sollte der Zähler den anzeigbaren Wertebereich verlassen, wird er auf 0 zurückgesetzt.
- Dokumentieren Sie im Quellcode aus welchem Grund die Interrupts aktiviert bzw. deaktiviert werden können/müssen.

Hinweise:

- Nutzen Sie die `libspicboard` sowohl zur Ansteuerung von Siebensegmentanzeige und LEDs als auch zum Auslesen des Potentiometer-Wertes. Die Verwendung der Button-Funktionen der `libspicboard` ist jedoch **nicht zulässig**.
- Um die Wahrscheinlichkeit, dass Tastendrücke verloren gehen, zu minimieren, dürfen die Interrupts während des Aufrufs der `sb_led_*`-, `sb_7seg_*`- und `sb_adc_*`-Funktionen nicht gesperrt sein.
- Verwenden Sie das Datenblatt um herauszufinden an welchen Interrupts die Taster (`PD2`, `PD3`) angeschlossen sind. Dort finden Sie auch die nötigen Informationen um das `EICRA` und `EIMSK`-Register richtig zu konfigurieren.
- Interrupts und Funktionen im Interruptkontext (Alarme) sollen, wie immer, möglichst kurz gehalten werden.
- Die Funktion `sb_timer_cancelAlarm()` erwartet einen Pointer vom Typ `ALARM *`. Sollte Sie jedoch einen Pointer vom Typ `volatile ALARM *` benötigen, erzeugt der Compiler folgende Warnung:
  `passing argument 1 of 'sb_timer_cancelAlarm' discards 'volatile' [...] [-Wdiscarded-qualifiers]`
  Sie können dies verhindern, in dem Sie den Pointer beim Aufruf entsprechend casten:
  `sb_timer_cancelAlarm((ALARM *) alarm);`
- Begründen Sie die Verwendung von allen `volatile` Variablen. Wenn für mehrere Variablen die selbe Begründung gilt, dürfen Sie diese gemeinsam begründen.
- Im Verzeichnis `/proj/i4spic/pub/aufgabe2/` befindet sich die Datei `izaehler.elf`, welche eine flashbare Beispielimplementierung enthält, mit der Sie die gewünschte Verhaltensweise des Programms sehen können.

Abgabezeitpunkt

alle Gruppen 22.11.2020 18:00:00