SPiC-Aufgabe #5: solaranlage

(15 Punkte, in Zweier-Gruppen)

Entwerfen Sie eine Steuerung für ein Kontrollpaneel einer Solaranlage in der Datei `solar.c`. Das Paneel soll im Normalzustand ausgeschaltet sein und keine Werte anzeigen. Möchte ein Nutzer die aktuell erzeugte Leistung einsehen, kann er das Paneel per Tastendruck aktivieren. Durch die Aktivierung wird die 7 Segment Anzeige aktiviert und zeigt die aktuelle Leistung der Solaranlage (PHOTO) an. Möchte der Nutzer die Art der Anzeige ändern, sorgt ein weiterer Knopfdruck für einen Wechsel in den LED Modus. Ist eine gewisse Zeit ohne Eingaben des Users vergangen, schaltet sich das Paneel wieder in den initialen Zustand ohne Anzeige. Die Aufgabe ist in zwei Teile unterteilt, im zweiten Teil soll zusätzlich zu der oben beschriebenen Logik eine Differenzierung zwischen kurzen und langen Aktivierungen von Button1 stattfinden. Um Probleme der Anlage zu vermeiden, soll ein langes Drücken von Button1 einen Reset simulieren und das Kontrollpaneel neu starten.

Teil a) Steuerung des Kontrollpaneels (11 Punkte)

Die Anzeige des Paneels wird durch die 7 Segment Anzeige dargestellt. Der LED Modus verwendet die LED Leiste und `sb_led_showLevel()`, um die relative Leistung der Anlage anzuzeigen. Durch das Drücken von `BUTTON0` wird das Paneel aktiviert und die 7 Segment Anzeige zeigt die relative Leistung der Solaranlage an. Hierfür soll der Photosensor `PHOTO` in regelmäßigen Abständen von 1 Sekunde ausgelesen werden und der Wert durch den maximalen Wert des Photosensors dividiert werden. Das Drücken von `BUTTON1` soll im aktivierten Zustand zwischen der Anzeige auf der 7 Segment Anzeige und der LED Leiste wechseln. Für letztere Anzeigeart soll der relative Wert des Photosensors als Füllstandsanzeige interpretiert werden. Erfolgt für 15 Sekunden keine Aktivierung eines Buttons wird das Paneel ausgeschaltet.

Das Paneel soll im Detail wie folgt arbeiten:

- Im Initialzustand ist das Kontrollpaneel ausgeschaltet und soll ausschließlich auf das Drücken von `BUTTON0` warten.
- Wird `BUTTON0` gedrückt, startet die Anlage, indem für eine Sekunde das Logo des Herstellers angezeigt wird, hier dargestellt durch den String "So" auf der 7 Segment Anzeige mit `sb_7seg_showString()`.
- Nach dieser Sekunde wechselt das Paneel in den angeschalteten Zustand mit Anzeige über die 7 Segment Anzeige. In Abständen von einer Sekunde soll der Wert des Photosensors, `sb_adc_read()`, ausgelesen werden. Der gelesene Wert soll dann auf das Intervall von 0 bis 99 abgebildet werden, sodass die 7 Segment Anzeige einen relativen Wert der Leistung der Solaranlage anzeigen kann.
- Wird für 15 Sekunden `BUTTON1` nicht gedrückt, schaltet sich die Anlage automatisch aus und geht in den Initialzustand über.
- Das Drücken von `BUTTON1` in beiden angeschalteten Zuständen soll zwischen diesen Wechseln.
- Im LED Modus soll analog zum 7 Segment Modus, jede Sekunde der Wert ausgelsen und diesmal mit `sb_led_showLevel()` auf dem LED Streifen abgebildet werden.

Teil b) Reset Funktionalität (4 Punkte)

- Im folgenden Teil wird eine Möglichkeit zum Reset des Kontrollpaneels implementiert. Dazu muss unterschieden werden können, ob `BUTTON1` kurz oder lang gedrückt wird.
- Um dies im Rahmen der Zustandsautomaten zu implementieren, sollen deshalb zwei Übergangszustände, jeweils vom 7 Segment Modus in den LED Modus und umgekehrt, zu dem existierenden Zustandsautomat hinzugefügt werden.
- Sobald der Button gedrückt wird, kann dann in den entsprechenden Übergangszustand gewechselt werden. Ein Verlassen des Übergangszustandes kann auf zwei Arten passieren:
  - Das Loslassen des Buttons sorgt dafür, dass der Modus gewechselt wird.
  - Vergeht eine Sekunde, ohne dass der Button losgelassen wird, so begibt sich das Kontrollpaneel zurück in den Zustand, in dem nur das Logo für eine Sekunde gezeigt wird.
  - Es muss dementsprechend darauf geachtet werden, dass unnötige Interrupts danach ignoriert werden.

Achten Sie darauf, dass der Mikrocontroller in Ruhephasen, in denen keine Berechnungen durchgeführt werden, in den Schlafmodus wechselt. Dies geschieht durch die entsprechenden Funktionen in `avr/sleep.h`.

Achten Sie weiterhin auf die korrekte Verwendung des `volatile`-Schlüsselworts. Beschreiben Sie in einem Kommentar zu jeder verwendeten `volatile`-Variable, weshalb Sie dieses Schlüsselwort dort benötigen.

Hinweise:

- Eine Abbildung auf einen Zustandsautomaten kann sehr hilfreich sein. Überlegen Sie dazu, welche Zustände es gibt, und wie zwischen diesen gewechselt werden kann.
- Verwenden Sie die Module `led` und `7seg` der `libspicboard` für die Ausgabe.
- Verwenden Sie *weder* das `button` *noch* das `timer` Modul der `libspicboard`!
  - Konfigurieren Sie stattdessen direkt die Interruptbehandlung und die Interrupthandler für `BUTTON0` und `BUTTON1`; diese sind an den Pins `PD2` (`BUTTON0`) bzw. `PD3` (`BUTTON1`) und damit an den externen Interruptquellen `INT0` bzw. `INT1` des ATmega-Mikrocontrollers angeschlossen. Bedenken Sie jedoch die ggf. unterschiedliche Interrupterkennungskonfiguration.
  - Für die Zeittaktung soll der `TIMER0` verwendet werden. Es darf die Überlaufunterbrechung `OVF` verwendet werden (Fehler bis einschließlich 50 ms sind tolerierbar). Es soll ein möglichst ressourcenschonender Vorteiler (`prescaler`) gewählt werden.
- Bauen Sie Ihr Programm so auf, dass in der `main()` Funktion nur an einer zentralen Stelle die Programmlogik zum Schlafen der CPU implementiert ist. Insbesondere soll die Programmlogik zum Schlafen nicht für jeden Zustandsübergang extra implementiert oder aufgerufen werden.
- Begründen Sie die Verwendung von allen `volatile` Variablen. Wenn für mehrere Variablen die selbe Begründung gilt, dürfen Sie diese gemeinsam begründen.
- Treffen Sie keine Annahmen über den initialen Zustand der Hardware.
- Im Verzeichnis `/proj/i4spic/pub/aufgabe5/` befindet sich die Datei `solar.elf`, welche eine Beispielimplementierung enthält.

Abgabezeitpunkt

Per Skript im CIP abfragbar: `/proj/i4spic/bin/get-deadline aufgabe5 Txx`