GSPiC-Aufgabe #5: Ampel

(15 Punkte, keine Gruppen)

Entwerfen Sie eine Steuerung für eine Bedearfsampel an einem Fußgängerüberweg in einer Datei `ampel.c`. Hierbei wird die den Autos zugewandte Ampel durch die LEDs `RED0`, `YELLOW0` und `GREEN0` dargestellt, die Fußgängerampel durch die LEDs `RED1` und `GREEN1` (kein gelbes Licht). Durch das Drücken von `BUTTON0` können Fußgänger eine Umschaltung anfordern. Die LED `BLUE1` signalisiert den Fußgängern, dass eine Umschaltanforderung entgegengenommen wurde.

Die Steuerung soll im Detail wie folgt arbeiten:

- Im Ausgangszustand zeigt die Autoampel grün, die Fußgängerampel rot.
- Eine Umschaltanforderung wird durch Druck auf `BUTTON0` ausgelöst. Der Druck der Taste wird durch Aktivierung der LED `BLUE1` bestätigt (entspricht „Signal kommt“). Diese LED wird wieder deaktiviert sobald die Fußgängerampel grünes Licht zeigt. Weitere Tastendrücke werden ignoriert, bis die Autoampel wieder grün zeigt (nicht aktiv).
- Nach erfolgter Umschaltanforderung zählt die Ampel über die Siebensegmentanzeige 8 Sekunden herunter, welche die Fußgänger noch warten müssen, bis ihre Ampel grün wird; in den übrigen Phasen bleibt die Siebensegmentanzeige aus. Von den insgesamt 8 Sekunden bleibt die Autoampel noch 5 Sekunden grün, dann wechselt sie für 1 Sekunde in den Zustand gelb, bevor sie schließlich rot wird. Erst nach weiteren 2 Sekunden, in denen beide Ampeln rot sind, schaltet die Fußgängerampel auf grün.
- Die Grünphase der Fußgängerampel soll exakt 5 Sekunden andauern, bevor sie wieder auf rot wechselt. Nach einer weiteren Sekunde wechselt die Autofahrerampel für eine Sekunde auf gelb-rot und wieder auf grün in den Ausgangszustand.

Achten Sie darauf, dass der Mikrocontroller in Ruhephasen, in denen keine Berechnungen durchgeführt werden, in den Schlafmodus wechselt. Dies geschieht entweder implizit, z. B. in `sb_timer_delay()` (siehe Dokumentation zu dieser Funktion), oder explizit durch die entsprechenden Funktionen in `avr/sleep.h`.

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

Hinweise:

- Verwenden Sie die Module `LED` und `7SEG` der `libspicboard` für die Ausgabe, sowie das Modul `Timer` für die zeitkritischen Aktionen.
- Verwenden Sie *nicht* das Modul `Button` der `libspicboard`! Konfigurieren Sie stattdessen direkt die Interruptbehandlung und den -handler für `BUTTON0`; dieser ist angeschlossen am Pin `PD2` und an der externen Interruptquelle `INT0` des ATmega32.
- Falls Sie die Alarm-Callback-Funktionen des `Timer`-Moduls verwenden, beachten Sie, dass diese im Kontext einer Unterbrechungsbehandlungsfunktion (ISR) ausgeführt werden! Achten Sie deswegen darauf, diese Funktionen, genau wie die Button-ISR, sehr kurz zu halten. (In keinem Fall soll die komplette Ampelschaltung in diesen ISRs realisiert werden!)
- Im Verzeichnis `/proj/i4gspic/pub/aufgabe5/` unter Linux bzw. in `S:\aufgabe5\` unter Windows befindet sich die Datei `ampel.hex`, welche eine Beispielimplementierung enthält.
- Ihr Programm muss mit der `Build`-Compiler-Konfiguration kompilieren und funktionieren; diese Konfiguration wird zur Bewertung herangezogen.

Abgabezeitpunkt

T01	Sonntag,	29.06.2014	18:00	T06, T07	Freitag,	27.06.2014	18:00
T02, T03	Montag,	30.06.2014	18:00	T08, T09, T10	Freitag,	27.06.2014	18:00
T04, T05	Dienstag,	01.07.2014	18:00				