Aufgabe 4: Ampel (15 Punkte, 2er-Gruppen, Abgabe bis Mi. 17.06.09 10:00 Uhr)

In dieser Aufgabe soll eine Ampelsteuerung in einer Datei **ampel.c** entwickelt werden. Gegeben ist das Szenario einer Fußgängerampel (FGA, *Red1*, *Green1*), die bei Bedarf aktiviert werden kann und dann für eine gewisse Zeit den Kfz-Verkehr über die Kfz-Ampel (KA, *Red0*, *Yellow0*, *Green0*) stoppt, um Fußgängern die Überquerung der Straße zu ermöglichen. Die Steuerung soll sich wie folgt verhalten:

- Im Ruhebetrieb zeigt die KA *grün*. Der Mikrokontroller soll im Ruhebetrieb im Idle-Powersave-Modus schlafen und aus diesem beim Eintreffen von Ereignissen durch Interrupts geweckt werden.
- Über den Taster *BUTTON0* können Fußgänger einen Umschaltwunsch signalisieren. Eine akzeptierte Umschaltanforderung wird durch Aktivierung der LED *BLUE0* bestätigt.
- Die minimale Dauer einer KA-Grünphase beträgt 5s. Wird *BUTTON0* nach deren Ablauf gedrückt, so startet die Umschaltung sofort, ansonsten erst nach Ablauf der Minimaldauer. *BLUE0* wird zu Beginn des Umschaltvorganges wieder deaktiviert.
- Beim Umschaltvorgang schaltet die KA über gelb auf rot um, anschließend wird die FGA auf grün geschaltet. Nach einer Dauer von 10s schaltet die FGA zurück auf rot und die KA über gelb-rot zurück in den Zustand grün. Die Steuerung wechselt wieder in den Ruhebetrieb. Zwischen den einzelnen Schaltübergängen sollen Wartezeiten von 1s bestehen.
- Die Ampel nimmt Tastendrücke erst wieder entgegen, nachdem die Rotphase beendet wurde (KA-Zustand rot-gelb). Vorherige Tastendrücke werden ignoriert.
- Während des Ablaufens der min. KA-Grünphase (5s) sowie der FGA-Grünphase (10s) wird die noch übrige Zeit in Form eines sekündlichen Countdowns auf den 7-Segment-Anzeigen dargestellt. Nach Ablauf eines Countdowns wird die 7-Segment-Anzeige deaktiviert. Zwischen Anzeigenaktualisierungen soll der Mikrokontroller in den Idle-Powersave-Modus versetzt werden.

Hinweise:

- Verwenden Sie zur Ansteuerung von 7-Segment-Anzeigen, LEDs sowie dem Timer die SPiCboard-Bibliothek. **Das Button-Modul darf nicht verwendet werden.** Hardwarekonfiguration sowie Interruptbehandlung für den Taster sollen selbst implementiert werden.
- Sie können die Funktion `sb_timer_delay()` zur Realisierung der 1s-Wartezeiten im Schaltvorgang verwenden. Die Funktion darf allerdings nicht während der Ruhephase oder des Countdowns verwendet werden. Verwenden Sie hier die Funktion `sb_timer_setAlarm()` um Interrupts zu erhalten, die den Mikrokontroller aus dem Schlafmodus wecken.
- Die Unterbrechungsbehandlungsfunktionen (ISR) sollen möglichst kurz gehalten werden - auf keinen Fall soll die komplette Ampelschaltung in einer ISR realisiert werden. Callback-Funktionen an das Timer-Modul sind mit ISRs gleichzusetzen.
- Achten Sie auf die korrekte Verwendung des **volatile**-Schlüsselworts, insbesondere bei Variablen, die von Interrupt und Hauptprogramm nebenläufig benutzt werden. Probleme können auch bei der aktiven Warteschleife entstehen, wenn das **volatile**-Schlüsselwort nicht korrekt verwendet wird. Verwenden Sie **volatile** nicht unnötig und versehen Sie jede Stelle, an der Sie eine **volatile**-Deklaration verwendet haben, mit einem Kommentar, mit dem Sie den Grund hierfür beschreiben.
- Ihr Programm soll mit der Optimierungsstufe **-Os** des GNU Compilers funktionieren. Falls Sie ein durch ein fehlendes **volatile**-Schlüsselwort verursachtes Problem vermuten, können Sie zum Test Ihr Programm ohne Compileroptimierungen (**-O0**) kompilieren.
- Achten Sie auf die korrekte Synchronisation von Hauptprogramm und Unterbrechungsbehandlungen. Stellen Sie sicher, dass ein Fußgänger, der den Taster nur ein einziges Mal betätigt, in jedem Fall die Ampel überqueren kann, auch dann, wenn keine weiteren Fußgänger den Taster betätigen. Kommentieren Sie jede Verwendung von **cli()/sei()** indem Sie das Problem beschreiben, welches den Einsatz der Synchronisationsprimitive an der jeweiligen Stelle erforderlich macht.