Aufgabe 4: Ampelsteuerung (15 Punkte, Abgabe bis Mi., 11.06.08 16:00)

Die Aufgabe kann in 2er-Gruppen bearbeitet werden. Die Abgabe erfolgt durch einen Bearbeiter.

In dieser Aufgabe soll eine Ampelsteuerung für das bereits bekannte AVR-Board entwickelt werden. Gegeben ist das Szenario einer Fußgängerampel, die bei Bedarf aktiviert werden kann und dann für eine gewisse Zeit den Autoverkehr stoppt, um Fußgängern die Überquerung der Straße zu ermöglichen. Hierbei sollen die drei einer Ampel entsprechenden LEDs auf dem Board der dem Kfz-Verkehr zugewandten Ampel entsprechen (Port B, Pins 0-2), die Fußgängerampel selbst wird nicht dargestellt. Die Ampel soll sich wie folgt verhalten:

- Im Ruhebetrieb zeigt die Kfz-Ampel *grün*, bis der Taster (Port D, Pin 2) betätigt wird. Der Mikrokontroller soll im Ruhebetrieb im Idle-Powersave-Modus sein und aus diesem durch den Druck auf den Taster aufgeweckt werden. Der Taster soll hierbei einen Interrupt auslösen und kann nicht mehr durch Polling abgefragt werden.
- Nach Druck des Tasters signalisiert die Ampel das kommende Signal auf der blauen LED (Port D, Pin 7). Der Umschaltvorgang beginnt nach einer Wartezeit von etwa 5 Sekunden. Die blaue LED wird vor Beginn des Umschaltvorganges wieder deaktiviert.
- Nach Ablauf der Wartezeit soll die Ampel über gelb auf rot umschalten. Nach einer Rotphase von etwa 10 Sekunden schaltet die Ampel über gelb-rot zurück in den Zustand grün und wechselt wieder in den Ruhebetrieb. Zwischen den einzelnen Schaltübergängen (gelb/rot, rot-gelb/grün) sollen Wartezeiten von etwa 1 Sekunde bestehen.
- Die Ampel nimmt Tastendrücke erst wieder entgegen, nachdem die Rotphase beendet wurde. Vorherige Tastendrücke sollen nicht beachtet werden.

Hinweise:

- Verwenden Sie keinen pegelgesteuerten Interrupt, da dieser im Simulator in Kombination mit HapSIM nicht richtig funktioniert. Die Unterbrechungsbehandlung soll möglichst kurz gehalten werden - auf keinen Fall soll die Unterbrechungsbehandlung die komplette Ampelschaltung realisieren.
- Wartezeiten sollen zunächst wie in der letzten Aufgabe durch aktive Warteschleifen realisiert werden, deren Länge experimentell zu bestimmen ist.
- 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 (Voreinstellung in AVR Studio). 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 Unterbrechungsbehandlung. 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.