Allgemeine Hinweise zu den SPiC-Übungen

- Die Aufgaben sind alleine zu bearbeiten.
- Alle an der Übung Teilnehmenden erhalten für SPiC jeweils ein spezielles Projektverzeichnis mit dem Namen `/proj/i4spic/<login>/`, wobei `<login>` für den eigenen Login-Namen steht. Die Projektverzeichnisse werden für alle Teilnehmer angelegt, die sich im Waffel-System angemeldet haben. Eine Anmeldung im Waffel-System ist daher zwingend zur Übungsteilnahme erforderlich! Die Projektordner werden auch in die SPiC-IDE eingebunden.
- Der Verzeichnisbaum für die Aufgaben ist folgendermaßen aufzubauen:
  `/proj/i4spic/<login>/aufgabe1`
  `/proj/i4spic/<login>/aufgabe2`
  ...
- Die Aufgaben sind bis spätestens zum Abgabetermin durch Aufruf des Programms
  `/proj/i4spic/bin/submit aufgabeX`
  (mit `X` = `1` ... `n`) abzugeben. Dieses Programm kopiert die in der Aufgabenstellung verlangten Dateien aus dem entsprechendem Verzeichnis. Bis zum Abgabetermin kann ein Programm beliebig oft abgegeben werden – es gilt der letzte, vor dem Abgabetermin vorgenommene Aufruf des Abgabeprogramms. Alternativ kann auch aus der SPiC-IDE abgegeben werden.
- Um seine letzte (und damit gültige) Abgabe zu überprüfen, kann die Abgabe per
  `/proj/i4spic/bin/show-submission aufgabeX`
  betrachtet werden. Um nur die Unterschiede zwischen der letzten Abgabe und dem aktuellen Stand im Projektverzeichnis anzuschauen, kann die Option `-d` übergeben werden.
  `/proj/i4spic/bin/show-submission -d aufgabeX`
- Die Abgabezeitpunkte sind abhängig von der eigenen Tafelübung. Ihren eigenen Abgabezeitpunkt können Sie durch die SPiC-IDE abfragen oder durch Aufruf des Programms
  `/proj/i4spic/bin/get-deadline aufgabeX`
- Eine genaue Beschreibung der SPiC-IDE und ihrer Funktionen ist unter `https://www4.cs.fau.de/Lehre/WS19/V_SPIC/SPiCboard/cip.shtml` zu finden.
- Eine Wertung bei Abgabe nach dem Abgabezeitpunkt kann **nur in begründeten Ausnahmefällen** nach Rücksprache mit Ihrem Übungsleiter erfolgen, der dann entscheidet, ob die verspätete Abgabe noch gewertet wird. Eine frühere, fristgerechte Abgabe wird durch eine verspätete Abgabe *nicht* überschrieben und im Zweifelsfall gewertet.
- Verwenden Sie für den Namen der C-Quelldatei, soweit in der Aufgabenstellung nicht anders angegeben, den Namen des Programms entsprechend dem Titel der jeweiligen Aufgabenstellung. Ist der Titel der Aufgabenstellung also z. B. *blink*, so legen Sie den Quellcode in einer Datei `blink.c` ab.
- Weitere Informationen finden Sie auf den Vorlesungsseiten:
  `https://www4.cs.fau.de/Lehre/WS19/V_SPIC/`
- Die Dokumentation der `libspicboard` finden Sie ebenfalls auf der Vorlesungsseite:
  `https://www4.cs.fau.de/Lehre/WS19/V_SPIC/SPiCboard/libapi.shtml`
- Details zur Entwicklungsumgebung sind hier zu finden:
  `https://www4.cs.fau.de/Lehre/WS19/V_SPIC/SPiCboard/cip.shtml`
- Eine Anleitung um den SPiCsim zu verwenden finden Sie hier:
  `https://www4.cs.fau.de/Lehre/WS19/V_SPIC/SPiCboard/spicsim.shtml`

SPiC-Aufgabe #1: zaehler

(6 Punkte, keine Gruppen)

Erstellen Sie ein Programm `zaehler` in einer Datei `zaehler.c`. Dieses Programm soll einen einfachen Zähler auf dem SPiCboard implementieren, der einen Zahlenwert automatisch in einer vom Potentiometer abhängigen Geschwindigkeit hochzählt und jeweils den aktuellen Wert mit Hilfe der LED-Reihe und der 7-Segmentanzeige darstellt.

Im Einzelnen soll das Programm wie folgt funktionieren:

1. Die 7-Segmentanzeige stellt dabei immer wieder aufsteigend die Werte von `0`–`99` dar, während für jede volle Hundert eine weitere LED eingeschaltet wird.
2. Wird der maximal darstellbare Zahlenwert erreicht, so beginnt der Zähler erneut bei `0`.
3. Die Geschwindigkeit des Zählers ist über das Potentiometer einstellbar (`sb_adc_read()`).

Unterteilen Sie dieses Problem zunächst in einzelne Teilprobleme. Überlegen Sie sich einen Ablaufplan dieser Teilprobleme und die entsprechenden C-Kontrollkonstrukte zu bedingter und wiederholter Ausführung, mit deren Hilfe Sie diesen Ablaufplan realisieren können.

Die Wartezeit zwischen den Schritten des Zählers soll in einer eigenen Funktion

```
static void wait(void);
```

realisiert werden. Diese ermittelt in linearer Abhängigkeit zur aktuellen Einstellung des Potentiometers die Wartedauer, die dann in einer aktiven Warteschleife realisiert wird.

Schreiben Sie außerdem eine Funktion

```
static void show_number(uint16_t num);
```

welche den aktuellen Zählerwert unter Verwendung der 7-Segmentanzeige und der LED-Reihe wie oben beschrieben darstellt. Der übergebene Parameter muss dabei entsprechend in die jeweiligen Bestandteile zerlegt werden. Sie dürfen Ihr Programm nach Wunsch in weitere Funktionen unterteilen.

Überlegen Sie sich bei den verwendeten Variablen, welche Lebensdauer und Sichtbarkeit diese benötigen und wählen Sie jeweils die kürzest nötige Lebensdauer und die geringst mögliche Sichtbarkeit für Ihre Variablen. Sie benötigen keine globalen Variablen in Ihrem Programm.

Hinweise

- Im Verzeichnis `/proj/i4spic/pub/aufgabe1/` befindet sich die Datei `zaehler.elf`, welche eine flashbare Beispielimplementierung enthält, mit der Sie die gewünschte Verhaltensweise des Programms sehen können.

Abgabezeitpunkt

T01	28.10.2019	18:00:00
T02	29.10.2019	18:00:00
T03	31.10.2019	18:00:00