Name	Matrikel	Anmerkungen
Datum	Raster (z.B. Mo-2x)	Testat/Datum

Legende: V: Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Praktikum 2

Lernziele: PSoC Creator einrichten, Evaluierungs-Board anschließen und programmieren, Ein-Ausgabe über PC-Terminal. Uart, I/O-Peripherie. C-Schlüsselwort volatile
Vorbereitung: PSoC Creator [1] installieren, TeraTerm [2] installieren. FreeSoC2 Dokumentation [3] bereithalten und sichten, siehe auch [4]..

Für die Bearbeitung der Aufgaben ist ein Termin angesetzt.

1. Starten Sie ***PSoC-Creator*** und laden Sie das Projekt Termin 2: MPS21_Prakt_2.
   a. Schließen Sie das *FreeSoC2*-Entwicklungsboard an und warten Sie ab bis alle Treiber installiert sind. (kann etwas dauern).
   b. Starten Sie ***TeraTerm*** und verbinden Sie mit `COM<xy>` (Datei > Neue Verbindung >Seriell)
      Es ist der COM<xy>: KitProg USB-UART(COM<xy>)
   c. Einstellungen > Serieller Port: baud: 115200, data: 8, parity: none, stop: 1
   d. Einstellungen > Terminal Übertrage: CR, Absenden: CR+LF
   e. Projekt MPS21_Prakt_2.zip entpacken. Doppelclick auf *MPS21_Prakt_2.cyprj*
   f. PSoC Creator: Build > Generate Application. (muss fehlerfrei sein)
   g. Creator: Build <your project> (muss fehlerfrei sein)
   h. *(geht nur mit Board!)* Creator: Debug > Program. Program wird auf Board übertragen, TeraTerm-Fenster sollte Begrüßungstext anzeigen.

2. Erzeugen Sie eine einfache Bildschirmausgabe:
   Formatierte Ausgabe (von Zahlen, char, strings, …)
   Hinweis: es hat sich als ressourcenschonender für den PSoC erwiesen, statt des bekannten printf("<fmt>", var, […]);
   eine formatierte Ausgabe in einen genügen großen string **`buffer`** zu machen.
   `sprintf( buffer, ( "<fmt>", var, […] );` und diesen dann mit
   `UART_PutString( buffer );` auszugeben

3. Vorbereitung: Betrachten Sie *TopDesign*. Fenster zeigt Schaltplan mit Komponenten: UART (Pins 2[0] und 2[1] für RX und TX), vier LEDs (Pin_N_R 2[7], Pin_N_Y 6[2], Pin_N_G 6[3] und Pin_E_CW 15[5]) als Output und einen Button (Pin_CWEW 6[5]) als Input.
   a. Überlegen Sie, wann die LEDs R, Y, G leuchten? Active high oder low?
   b. Überlegen Sie, wann LED_CW leuchtet? Transistor Q_2 wirkt als Inverter!
   c. Welchen Pegel nimmt Pin_CWEW an, wenn SW_7 bzw. SW_8 nicht gedrückt ist und wenn gedrückt ist.

4. Lesen vom Zeichen vom Terminal, Menu, Schalten der LED(s) über Terminaleingaben
   a. Lesen Sie Zeichen von der UART: **`UART_GetChar()`**
   b. Schreiben Sie ein *einfaches* Menu, welches durch die Eingabezeichen gesteuert wird.
   c. Steuern Sie mit den Zeichen das Verhalten der LED(s) (s. 3, An- u. Ausschalten)

5. Taste CWEW abfragen und Aktion, toggeln der weißen LEDs überprüfen
   a. Wird nur unzuverlässig funktionieren, weil die Taste prellt!
      (*Prellen bedeutet, dass der mechanische Kontakt nicht sofort zuverlässig den neuen Zustand einnimmt, und anfangs mehrmals zurückspringt*)
   b. Wie könnte man per Software entprellen?

6. Aktivieren Sie das Blinken der roten LED durch Entfernen der Kommentarzeichen ‚//'.
   Funktion: *an … warten … aus … warten*.
   Das Warten wird mit Schleifen realisiert (*nicht ändern!*).
   a. Kompilieren Sie mit **Debug** und testen Sie Funktion.
   b. Kompilieren Sie jetzt mit **Release** und testen Sie Funktion. Warum geht es jetzt nicht mehr? (es scheint, als ob die Schleifen nicht ausgeführt werden)
   c. Erinnern Sie sich an Rechnerarchitektur und Optimierung des Codes durch den Compiler. (wird die Variabel ‚loop' denn gebraucht?)
      Recherchieren Sie in C-Referenzen nach ‚volatile'. Verwenden und testen Sie!

7. Testen Sie jetzt mit 6. nochmal die Funktion der Keyboard-Eingabe und des Buttons
   a. Sind Sie mit der Schnelligkeit der Reaktion auf Terminal-Eingabe zufrieden?
   b. Funktioniert der Button überhaupt noch zuverlässig?
   c. Warum ist das so? *Könnte man so eine Lösung erfolgreich verkaufen?*

8. Kommentieren Sie – gegebenenfalls nach dem Praktikum zu Hause – Ihren Code. Archivieren Sie Ihr Projekt zu Ihrem späteren Gebrauch.

9. Schreiben Sie ein kurzes Protokoll und fassen Sie Ihre Erkenntnisse zusammen und fügen Sie die jeweiligen Codeabschnitte hinzu. Laden Sie Ihren Code[+)] als *.zip und Ihr Protokoll als *.pdf in Moodle hoch bis **maximal** 1 Woche nach dem Termin.

Bereiten Sie sich auf den Praktikumstermin 2 so vor, dass die Zeit zur Durchführung während des Termins sicher ausreicht. *(Lesen Sie bitte die Aufgabenstellung und Begleitmaterial vor dem Praktikumstermin.)*

Die Themen und Erkenntnisse aus diesem Praktikum werden im Lauf des Semesters weiter benötigt! Arbeiten und dokumentieren Sie Ihre Ergebnisse sorgfältig!

Die Teilaufgaben sind schriftlich zu dokumentieren. Laden Sie Ihr Protokoll wie in 9. beschrieben zu Termins 2 hoch.

Viel Spaß und Erfolg

*+) im *.zip bitte **nur** den Ordner mit *.c, *.h und gegebenenfalls Projektdatei*.

[1] https://www.cypress.com/products/psoc-creator-integrated-design-environment-ide
[2] http://ttssh2.osdn.jp/index.html.en
[3] https://www.sparkfun.com/products/13714 [Documents]
[4] MPS-Skript zu dieser Vorlesung, Anhang C