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

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

Praktikum 5

Lernziele: Timer, Watchdog und Scheduler in GNU C auf PC (online-Praktikum)

Für die Bearbeitung der Aufgaben ist ein Termin angesetzt. **Eine gute Vorbereitung ist zwingend erforderlich!**

1. Testen Sie das Template. Beachten Sie die mit doxygen erzeugte Dokumentation [1] !
    a. Beobachten Sie die Wirkung des Watchdog (‚der über das Keyboard mit ‘r‘ zurückgesetzt werden kann).
    b. Kommentieren Sie dann erst einmal den Codebereich für den Watchdog aus.
    c. Die Timer-ISR sollte alle 1/10 s feuern. Testen Sie!
    Falls Sie nicht Windows benutzen, kann der System-Zeitstempel (clock()) von 1 ms abweichen. Versuchen Sie dann in myConf.h das andere Define MY_CORR_CLOCKS_PER_MS zu aktivieren und testen Sie erneut.

2. Mehrere kleine Tasks (Funktionen) mit unterschiedlichen Perioden (zeitlichen Häufigkeiten) ausgeführt werden. "schedule.h" gibt Vorgaben zu den Zuständen der Tasks: **`WAIT`** warten auf Ausführung, **`RDY`** ausführungsbereit, **`RUN`** wird ausgeführt.

    Der Typedef der Struktur **`sTskLst_t`** fasst die notwendigen Attribute für jeden Task zusammen. **`FpTsk_t fpTask`** ist der Zeiger auf eine Funktion vom Typ **`task_t`**.
    Die Periode, mit der der Task ausgeführt werden soll, ist **`period`**. Die Zeit, die der task auf Ausführung wartet, ist **`elapsed`**.
    a. Die (zu schreibende) Funktion **`schedule()`** (dt. „Planen“) überprüft den Zustand jedes Tasks, setzt die bereits gewartete Zeit hoch und falls **`period`** erreicht, den Zustand auf **`RDY`**.
    b. Die (zu schreibende) Funktion **`dispatch()`** (dt. „Ausführen“), in der Endlosschleife aufzurufen, führt alle **`RDY`**-Tasks aus und setzt danach den Status wieder auf **`WAIT`** und **`elapsed`** natürlich wieder auf 0.
    c. Ergänzen Sie weitere Tasks vom Typ **`task_t`** zum Testen.
    d. Verwalten Sie diese Tasks unter Benutzung von **`sTskLst_t`**.
    Hinweis: auch von Strukturen kann man ein Array anlegen …
    e. Nutzen Sie einen Timer (10 – 100 ms) um den Scheduler schedule() regelmäßig aufzurufen.

3. Dokumentieren Sie Ihren Code mit doxygen [2].
    a. a. Erzeugen und füllen Sie Dokumentationsblocks für jede Ihrer Funktionen.
    b. Hinweis: Nutzen Sie in CodeBlocks den Menupunkt DoxyBlocks.
    c. b. Betrachten Sie die erzeugte Dokumentation als .html zum Testen.
    d. c. Erzeugen Sie die Dokumentation durch entsprechende Einstellung in doxygen
    e. auch als .rtf oder .pdf (erfordert LaTeX, aber nicht verlangt!) und geben Sie
    f. diese mit dem Protokoll ab.

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

5. 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 5 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 5. beschrieben zu Termins 5 hoch.

Viel Spaß und Erfolg

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

[1] API-Dokumentation zum Framework (refman_MPS-Prakt_5.pdf)
[2] https://www.doxygen.nl/index.html
[3] https://graphviz.org/