1. Übung: Multithreading-Betrieb auf einem Prozessor der 8051-Familie **Vertiefung Echtzeitsysteme**

Abgabe:30. Mai 2013 (Veranstaltung 1 + 2)

Ausgehend von dem verteilten Beispiel-Code zur Thread-Umschaltung sind die folgenden Ergänzungen/Erweiterungen vorzunehmen:

- Mit einer anderen Timer-Betriebsart (z.B. Reload eines 16-bit Zählers in `timer0(...)` statt Auto-Reload mit 8-bit Zähler) ist die Zeit zwischen zwei Timer-Interrupts auf z.B. 10 ms zu verlängern.
- Jeder Thread erhält eine Priorität. Das Scheduling soll derart erfolgen, dass immer der rechenwillige Thread mit der höchsten Priorität in den Zustand „rechnend“ übergeht. Es darf zur Vereinfachung vorausgesetzt werden, dass alle Threads unterschiedliche Prioritäten haben. (Threads mit gleicher Priorität wären andernfalls sinnvoll nach dem Zeitscheibenverfahren zu verwalten.)
- ersten Thread als idle-Thread verwalten
- Die Anzahl der Threads darf auf drei (einschließlich idle-Thread) beschränkt sein
- `sleep(ms)` realisieren
- Semaphore realisieren (`seminit(...)`, `semwait(...)`, `sempost(...)`)
- Aufteilen des gesamten Codes in zwei Quelltext-Module:
  1) Code des MT-Betriebssystems,
  2) User-Code (frei von Interna des MT-Betriebssystems)
- Nachweis der Funktionalität durch die unten angegebene Test-Applikation. Diese muss auch auf der realen Hardware laufen.
- Der Code muss hinreichend kommentiert sein. Funktionen benötigen einen Kommentar-Kopf mit der Beschreibung ihrer Aufgabe und der Parameter (zusätzlich Angabe ob Eingangs-, Ausgangs- oder Durchgangsparameter). Vergessen Sie nicht die Angabe der Namen der Gruppenmitglieder im Programm-Kopf.

Achtung:

- Beim Anlegen des Projektes in µVision als Target „Infineon C515C-L“ wählen!
- Unbedingt unter Project -> Options for Target1 -> Target das „Memory Model“ Large wählen!
- "Lokale" Variablen in einem Thread oder einer von ihm aufgerufenen Funktion als `static` definieren!
- Verwenden Sie für Funktionen, die aus der Interrupt-Funktion, bzw. einer Thread-Funktion aufgerufen werden, mit "using n" immer die richtige Registerbank

Test-Applikation

Erstellen Sie in einer separaten Quellcodedatei ein Testprogramm, das diese Funktionalitäten durch ein dauerhaft laufendes System aus drei Threads (einschließlich idle-Thread) nachweist:

- Thread1 und Thread2 bilden ein Erzeuger/Verbraucher-Paar, die über einen Ringpuffer mit n Elementen (k long-Werte pro Ringpufferelement) Daten austauschen. Die Forderungen, dass der Erzeuger nichts in einen vollen Ringpuffer ablegen und der Verbraucher nichts aus einem leeren Ringpuffer entnehmen darf, sind durch Semaphore sicher zu stellen. Die long-Werte werden vom Erzeuger fortlaufend hochgezählt. So kann der Verbraucher überprüfen, ob er Werte in falscher Reihenfolge erhält. Sollte dieser Fehlerfall eintreten, so ist der Pegel auf Port5-Bit1 (Ruhezustand low) auf high-Pegel zu setzen.
- Thread1 besitze die höchste Priorität. Er soll nach jeweils 10 von ihm gefüllten Ringpufferelementen für 20 ms Port5-Bit0 auf high-Pegel setzen (low-Pegel sei der Ruhezustand).