

Übung 1 RTO

Vorläufer von Betriebssystemen, kooperative Task-Weitergabe:

Allgemeine Hinweise und benötigte Unterlagen:

- **Debugging-Werkzeuge (IDE):**
 - o Watch Windows, Memory Windows, SystemViewer → CorePeripherals
- **Dokumentation:**
 - o User Manual des APUS-Boards
 - o Datenblatt STM32F0xxx (Cortex-M0)
 - o Cortex-M0 Programming Manual (ARM)
- **Messhardware:** Saleae Logic Analyzer

Übungsaufgabe A – Implementierung einer Debug-Unit zur Timing-Messung mit dem Saleae-Logic-Analyze

Implementieren Sie eine Debug-Unit, die mit einem Saleae-Logic-Analyzer präzise Laufzeitmessungen von Tasks und Interrupts ermöglicht.

- Legen Sie die Datei Debug.c an.
- Verwenden Sie die folgenden GPIO-Pins des Arduino-Steckers (Port B):

Pin	Signal	Beschreibung
PB0	SystemTick	Systemtick-Interrupt aktiv
PB1	TaskCounter	Task-Counter aktiv
PB3	TaskKey	Tasten-Task aktiv
PB12	TaskLed	LED-Task aktiv
PB13	TaskWatch	Watchdog-Task aktiv
PB14	TaskPoti	Potentiometer-Task aktiv
PB15	TaskMandelbrot	Mandelbrot-Task aktiv

Immer wenn ein Task oder der SystemTick-Interrupt startet, wird die entsprechende Leitung auf high gesetzt. Wenn der Task oder SystemTick-Interrupt endet, wird die Leitung auf Low gesetzt.

Im Anhang ist die Pin-Belegung des APUS-Boards dargestellt.

Teilaufgaben:

1. Implementieren Sie die Debug-Unit in Debug.c.
2. Messen Sie mit dem Saleae-Logic-Analyzer die Laufzeit jedes Tasks.
3. Dokumentieren Sie die Messergebnisse durch Screenshots der Logic-Analyzer-Auswertung.
4. Bestimmen Sie den Overhead der Messroutine (in CPU-Zyklen und Mikrosekunden) und geben Sie diesen an.

Übungsaufgabe B – Reaktionsgeschwindigkeit t in einer Superloop-Architektur:

Ausgangssituation: Die Übungsvorlage enthält eine Superloop mit den folgenden Tasks:

```
while (1) {  
    TaskCounter();  
    TaskKey();  
    TaskLed();  
    TaskWatch();  
    TaskPoti();  
    TaskMandelbrot();  
}
```

Die Tasks sollen wie folgt erweitert werden:

Task	Beschreibung
Counter	Gibt einen Zählerwert aus, der bei 0 beginnt und pro Aufruf um 1 inkrementiert wird.
Key	Liest die Tasten ein und gibt die Tastenwerte auf dem Display aus. Bei Druck der Taste „USER“ wechselt die Schriftfarbe von Blau auf Grün oder Schwarz.
Led	Realisiert ein Lauflicht an den LEDs Led3, Led4, Led5; jede LED leuchtet abwechselnd für 100 ms.
Poti	Zeigt den aktuellen Potentiometer-Wert auf dem Display an.
Watch	Zeigt die Betriebszeit im Format mm:ss (Minuten:Sekunden) an.
Mandelbrot	Zeichnet die Mandelbrot-Menge auf dem Display.

Aufgaben:

- 1. Implementierung:** Erweitern Sie die Tasks gemäß der obigen Beschreibung.
- 2. Echtzeit-Anforderungen:** Optimieren Sie die Superloop so, dass alle Tasks ihre Ausgaben innerhalb der folgenden Zeitgrenzen aktualisieren:
 - **Key und Poti:** ≤ 50 ms Reaktionszeit
 - **Watch:** ≤ 1 s Aktualisierungsintervall
 - **Counter:** ≤ 20 ms Zykluszeit
- 3. Verifikation:** Messen Sie die Laufzeit jedes Tasks mit der in Aufgabe A implementierten Debug-Unit und dokumentieren Sie die Ergebnisse.

Übungsaufgabe C – Verwendung von Process-Stack (PSP) und Main-Stack (MSP)

Aufgabenstellung:

Erweitern Sie Ihr Programm so, dass der Mikrocontroller sowohl den Process-Stack Pointer (PSP) als auch den Main-Stack Pointer (MSP) verwendet.

Analysieren Sie anhand des Datenblatts und des μ Vision-Debuggers den genauen Ablauf bei Interrupts: Welche Register und Daten werden zu welchem Zeitpunkt auf welchem Stack gespeichert?

Teilaufgaben:

1. **Implementierung:** Erweitern Sie das Programm aus Aufgabe B um die Dual-Stack-Verwendung (PSP und MSP).
2. **Stack-Layout:** Definieren Sie sinnvolle Speicherbereiche für den Process-Stack und den Main-Stack im RAM. Begründen Sie Ihre Wahl (z. B. Größe, Trennung von Interrupt- und User-Code-Stack).
3. **Analyse und Dokumentation:**
 - Erstellen Sie Screenshots der Stack-Inhalte im μ Vision-Debugger (vor/nach Interrupts).
 - Dokumentieren Sie anhand von Auszügen aus dem Datenblatt (Cortex-M0 Programming Manual), welche Register bei einem Interrupt automatisch auf welchem Stack gesichert werden.
 - Erläutern Sie die Funktion von PSP und MSP im Kontext von Interrupt-Behandlung und Task-Execution.

Hinweis: Relevante Informationen finden Sie im Cortex-M0 Technical Reference Manual (Kapitel zu Exception Entry/Return) und im STM32F0xxx Reference Manual (Memory Map).

Abgabe und Beurteilung:

Gesamtpunktzahl: 10 Punkte (für die Aufgaben A, B und C)

Abzuliefern:

- **Quellcode:** Komplettes Projektverzeichnis (inkl. Debug.c und aller Task-Implementierungen)
- **Code-Dokumentation:** Kommentierter Quellcode mit Funktionsbeschreibungen (z. B. Doxygen-Style oder aussagekräftige Kommentare)
- **Messdaten:** Screenshots der Saleae Logic Analyzer-Messungen für alle Tasks und den SystemTick
- **Ergebnisübersicht:** Tabellarische oder grafische Darstellung der gemessenen Laufzeiten (Tasks und SystemTick)
- **Analyse:** Dokumentierte Auswertung der Stack-Analyse (PSP/MSP) mit Screenshots aus µVision und Referenzen auf das Datenblatt

Bewertungskriterien:

- **Verständnis:** mündliche Abfrage zu Implementierungen, Messergebnissen und Analysen
- **Korrektheit:** Alle Tasks funktionieren wie spezifiziert; Zeitgrenzen werden eingehalten.
- **Vollständigkeit:** Alle geforderten Deliverables sind vorhanden.
- **Dokumentation:** Code ist kommentiert; Messergebnisse und Analysen sind nachvollziehbar dargestellt.
- **Form:** Übersichtliche Struktur, korrekte Schreibweisen, vollständige Dateien.

Achtung: Die Ergebnisse dieser Übung werden für nachfolgende Übungen benötigt.

Anhang:

Zuweisung Debug-Port-Pins APUS

