

RTOS

Features

Hoch optimiert für den Einsatz in Systemen mit begrenzten Ressourcen

Optimierter Speicherbedarf
Maximal 4 KB ROM
200 Byte RAM^(*)

Voll konfigurierbar

Event-Based Scheduling

2 Prioritätsebenen

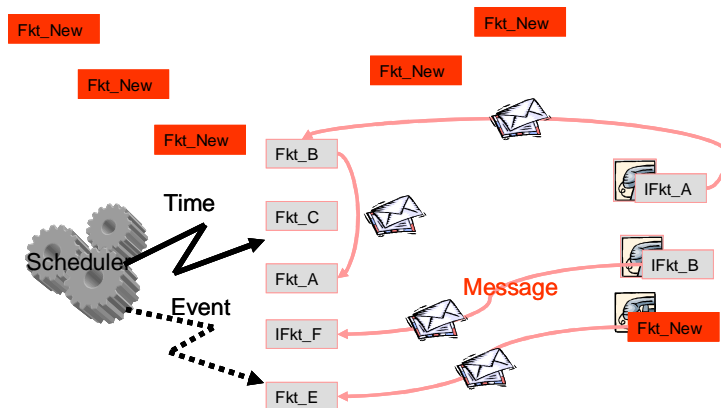
Optimierte Interrupt Latency je nach Zielplattform zwischen 0 und 6 Befehlszyklen^(*)

Optimiert für den Einsatz von UML. Kein Overhead durch Operating System Abstraction Layer.

Auslieferung mit Source Code

Keine Royalties

^(*) Abhängig von der Zielplattform



Wer nicht kommt zur rechten Zeit, der muss sehen, was übrig bleibt

so funktioniert der Scheduler des Embedded OO RTX.

Objektorientiert wird das Architektur Design über Events aufgebaut. Der Scheduler sorgt dafür, dass die auf das jeweilige Event wartenden Tasks Rechenzeit bekommen.

Auf diese Art wird ähnlich wie bei den MFC's (Microsoft© Foundation Classes) eine Laufzeit-Architektur bereitgestellt, die die Basis für ein robustes Software Design bietet.

Das Embedded OO RTX ist eines der ersten Betriebssysteme, das speziell für den Einsatz in Verbindung mit der UML entwickelt wurde. Es unterstützt in hoch optimierter Form alle notwendigen Betriebssystem-Dienste, die aus

Sicht der Notation UML erforderlich sind.

- Timer
- Events
- Messages
- Tasks

Auf die Implementierung von Diensten wie zum Beispiel Semaphoren oder Mutexes, wurde bewusst verzichtet. Ein Design, welches die objektorientierten Ansätze verfolgt, sollte auf alle Dienste verzichten die Tasks gegenseitig blockieren können. Aus dem Verzicht ergibt sich ein weiterer Vorteil, der Anwender wird bei optimalem Ressourcenbedarf zu einem OO Architektur-Design geführt.

Das wiederum führt automatisch zu einem Software Design mit besten Voraussetzungen für effiziente Wiederverwendung, Test und Teamarbeit.

Keep it structured

Nach diesem Motto empfehlen wir heutige Software Designs aufzubauen. Hier ist ein kleines Beispiel, wie die Basis-Architektur einer Applikation anstelle mit einer main() Loop mit dem Embedded OO RTX aussehen könnte.

```
#include "Blinky.h"

static void Blinky_entDef(void * const me)
{
    me->rootState_active = Blinky_OFF;
    RXF_schedTm(me->rxfr_reactive.myTask, 300);
}

static RXFTakeEventStatus Blinky_dispatchEvent(void * const
void_me, short id)
{
    RXFTakeEventStatus res = eventNotConsumed;
    switch (me->rootState_active)
    {
        case Blinky_OFF:
            if(id == Timeout_id)
            {
                RXF_unschedTm(me->rxfr_reactive.myTask);
                me->rootState_active = Blinky_ON;
                RXF_schedTm(me->rxfr_reactive.myTask, 200);
                res = eventConsumed;
            }
            break;
        case Blinky_ON:
            if(id == Timeout_id)
            {
                RXF_unschedTm(me->rxfr_reactive.myTask);
                me->rootState_active = Blinky_OFF;
                RXF_schedTm(me->rxfr_reactive.myTask, 300);
                res = eventConsumed;
            }
            break;
        default:
            break;
    }
    return res;
}
```

Die Vorteile liegen auf der Hand:

- Programmierung von Zeitabhängigkeiten ohne Hardware Timer und Interrupt Handling
- Immer gleiches Zeitverhalten bei verschiedenen Auslastungen des Systems (leichte Erweiterbarkeit)
- Objektorientierte Schnittstellen, keine Globalen Variablen als Kommunikation zwischen nebenläufigen Teilen des Systems. (Einfache Wiederverwendung und Erweiterbarkeit)
- und viele weitere

Diese Vorteile sind die Grundlage zur Bewältigung der heutigen Anforderungen an die meisten Embedded Applikationen und die Beherrschung wachsender Komplexität. Robustheit - Verstehbarkeit - Änderbarkeit - Wartbarkeit - Wiederverwendbarkeit.



Willert Software Tools GmbH

Tel. +49 5722 9678 60 - FAX: +49 5722 9678 80
Email: info@willert.de - www.willert.de
Hannoversche Str. 21 - 31675 Bückeburg

Steckbrief:

- 12 Monate Gewährleistung
- Lieferung inklusive Sourcecode.
- Keine Royalties
- Unterstützte Zielplattformen siehe www.willert.de
- Preis: 1.450,- Euro +MwSt

Voraussetzungen und Kenntnisse für den erfolgreichen Einsatz des Produktes:

- Erfahrungen im Einsatz von 'C' für Embedded Systeme (Sichere Anwendung von Compiler Linker)
- Ausreichend Kenntnisse über die eingesetzte Zielplattform ('C' Start-Up Konfiguration, Speicher Mapping)
- Kenntnisse im Umgang mit einem RTOS und im OO Architektur-Design. Ideale Voraussetzung ist die Teilnahme an unserem RTOS StartUp Training.
- ACHTUNG: Wir empfehlen dieses Produkt ohne Erfahrungen im OO Architektur Design von Laufzeit-Architekturen nur in Verbindung mit dem Embedded UML Studio® einzusetzen.

Rhapsody® is a registered trademark of Telelogic an IBM Company

Ihr lokaler Kontakt: