Embedded Realtime OS FreeRTOS auf STM32F4

Michael Ebert
Ad-hoc Networks GmbH
ebert@ad-hoc.network

Christoph Bläßer
christph.blaesser@gmx.de

Stichwörter
RTOS, ARM , STM32, Real Time.

KURZFASSUNG
Im Rahmen des vorliegenden Papers wird das Echtzeitbetriebssystem FreeRtos vorgestellt. Hierzu werden zu Beginn die allgemeinen Vorgaben für Echtzeitbetriebssysteme beschrieben. Im Verlauf des Textes wird an ausgewählten Beispielen dargestellt, wie FreeRos diese Anforderungen berücksichtigt und durch geeignete Programmfunktionen umsetzt.

1. ECHTZEITBETRIEBSSYSTEME

1.1 Vorteile eines Echtzeitbetriebssysteme

Mit der steigenden Leistungsfähigkeit von modernen uProzessoren, steigen auch die Anfordernungen an die Software die auf diese Systeme aufsetzt. Viele dieser Anwendungen verlangen trotz ihrer Komplexität, dass Teile des Programmablauf in bestimmten zeitlichen Grenzen ausgeführt wird und somit vorhersehbar und deterministisch sind. Es gibt prinzipiell drei verschiedene Modelle wie der Programmablauf einer embedded Anwendung gestaltet sein kann, siehe Abbildung 1. Anwendungen können in einer einzigen Schleife laufen oder aber in event-gesteuerten nebenläufigen eigenständigen Programmabschnitten (Thread oder Task) ausgeführt werden. Diese Nebenläufigkeit ist nur durch einen RTOS Kernel (Scheduler) möglich. Für viele kleine Anwendungen kann die Nutzung einer einzigen Schleife durchaus sinnvoll sein, sollten beispielsweise die Ressourcen so knapp sein, dass ein Overhead an Funktionalität ausgeschlossen werden muss. Ein großer Nachteil der „einschleifen Variante“ ist die permanente Nutzung des Prozessors. Besonders bei neuen akkubetrieben Geräten z.B. IoT Device wird sehr genau auf die Energieaufnahme geachtet. Ein RTOS bietet hingegen Funktionen mit denen sehr leicht ermittelt werden kann, ob ein Gerät in einen Schlafmodus wechseln kann, dies wird in Abschnitt 2.10 an Beispielen von FreeRtos und einem ARM uProzessor demonstriert.
TODO
Neben der Echtzeitfähig gibt es aber noch viele weitere Vorzüge für den Einsatz eines Echtzeitbetriebssystems. Durch das Herunterbrechen der Anwendung in Task entstehen viele kleine Module, die jeweils eine kleine Teilaufgabe des Gesamtsystems übernehmen. Die Anwendungen werden so skalierbar und erweiterbar. Die Entwicklungsarbeit lässt sich leicht auf Teams verteilen, die unabhängig ihre Module (Tasks) entwickeln und testen. Dies ermöglicht auch den Einsatz von agilen Entwicklungsmethoden wie Scrum in der embedded Enticklung. Abschnitt 3 Echtzeitanalyse detailliert erläutert.

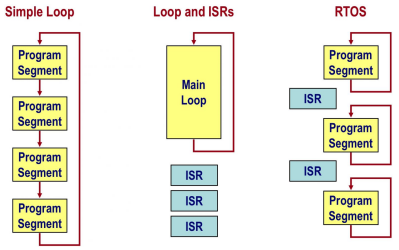

Abbildung 1. Übersicht Programmabläufe
Quelle: `http://www.embedded.com/`

RTOS aufsetzt, erfüllt somit nicht sofort Harten oder Weichen Echtzeitkriterien. Diese Kreterien zu erfüllen liegt beim Entwickler selbst, das Echtzeit Betriebssystem stellt nur die nötigen Werkzeuge zur Verfügung. Wie geprüft werden kann ob Echtzeitkriterien erfüllt werden, wird in Kapitel X näher erleutert.

2. FREERTOS

2.1 Geschichte

2.2 Zielsysteme

2.3 Entwicklungsumgebung

2.4 Ordner und Dateistrukturen

2.5 Scheduling

2.6 Memory Management

2.7 Interprozess Kommunikation

2.8 Software Timer

2.9 Interrupt Handling

2.10 Low Power Modes auf STM32F4 (ARM Cortex M3)

3. ECHTZEITANALYSE

4. DEBUGGING VON ECHTZEITSYSTEMEN

5. ECHZEIT PATTERN

6. CODING HINDERNISSE

7. ZUSAMMENFASSUNG

Literatur