Embedded Realtime on ARM uC - 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 weerden zu Beginn die allgemeinen Vorgaben für Echtzeitbetriebssysteme beschrieben. Im Verlauf des Textes wird an ausgewählten Beispielen dargestellt, wie freeRTos diese Anforderungen berücksichtigt und durch geeignete Programmfunktionen umsetzt. [1, 3].

1. ECHZEITBETRIEBSYSTEME

1.1 Vorteile eines Echzeitbetriebssystems

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 der Programmablauf der Software oder zumindest einige Teile dieser Software in bestimmten zeitlichen Grenzen ausgeführt wird und somit vorhersehbar und deterministisch ist. Ein Echzeitbetriebssystem (RTOS) bietet einem Entwickler die Möglichkeit ein solches System zu entwerfen. Eine Anwendung die auf ein Echzeitbetriebsystem aufsetzt, eerfüllt somit nicht sofort Harten oder Weichen Echzeitkriterien. Diese Kreterien zu erfüllen liegt beim Entwickler selbst, das Echtzeit Betriebsysstem stellt nur die nötigen Werkzeuge zur verfügung. Wie geprüft werden kann ob Echtzeitkriterien erfüllt werden, wird in Kapitel X näher erleutert. Neben der Echzeifähigkeit gibt es aber noch viele weitere Vorzüge für den Einsatz eines Echzeitbetriebssystems. Durch die Nebenläufigkeit etstehen viele kleine Module, die als seperate Anwendungen laufen (Hier ein Beispiel mit einem Board und Komponenten). Diese Module können in Teams entwickelt werden und eigenständigen Tests unterzogen werden. Ein häufiges Problem in embedded Systemen in die Problematik des Hardwarenahen Testens. Ein weiterer sehr wichtiger Punkt ist die Entwicklung von Energieeffizienten systemen. in einer anwendung die auf einem echzeitbetriebssystem aufsetzt, lässt sich durch die ereignisorientierte entwicklung sehr schnell feststellen, ob alle Aufgaben abgearbeitet wurden, so dass man ggf. das System schlafen legen kann. Wie so etwas in FreeRtos umgesetzt wurde und wie es für eine Beispielhafte Anwendung implementiert werden kann wird in Kapitel x detailiert erläutert.

2. FREERTOS

2.1 Geschichte

2.2 Zielsysteme

2.3 Entwicklungumgebung

2.4 Ordner und Dateistrukturen

2.5 Scheduling

2.6 Memory Management

2.7 Interprozess Kommunikation

2.8 Software Timer

2.9 Low Power Modes auf STM32F4 (ARM Cortex M3)

3. ECHTZEITANALYSE

4. DEBUGGING VON ECHZEITSYSTEMEN

5. ECHZEIT PATTERN

6. CODING HINDERNISSE

7. ZUSAMMENFASSUNG

Literatur

1. How to classify works using ACM's computing classification system. `http://www.acm.org/class/how_to_use.html`.
2. D. L. Black, D. B. Golub, K. Hauth, A. Tevanian, and R. Sanzi. The Mach exception handling facility. In *Proceedings of the 1988 ACM SIGPLAN and SIGOPS workshop on Parallel and distributed debugging*, PADD '88, pages 45–56, 1988.
3. M. Y. Ivory and M. A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516, 2001.