

Embedded Realtime on ARM uC - FreeRTOS auf STM32F4

Michael Ebert
Rotdornweg 28
Ad-hoc Networks GmbH
25451 Quickborn
ebert@ad-hoc.network

Christoph Bläßer
christph.blaesser@gmx.de

Stichwörter

RTOS, ARM, STM32, Real Time.

8. ZUSAMMENFASSUNG

1. EINFÜHRUNG ECHZEITBETRIEBSSYSTEME
2. VORTEILE EINES ECHZEITBETRIEBSSYSTEMS
3. FREERTOS GRUNDLAGE
 - 3.1 Ordner und Dateistrukturen
 - 3.2 Scheduling
 - 3.3 Interprozess Kommunikation
 - 3.4 Memory Management
4. FREERTOS IMPLEMENTIERUNG
5. ECHTZEITANALYSE
6. DEBUGGING VON ECHZEITSYSTEMEN

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 FreeRTOS diese Anforderungen berücksichtigt und durch geeignete Programmfunktionen umgesetzt. [1, 3].

7. ÜBERSICHT UND GRÜNDE FÜR DEN EINSATZ EINES RTOS

Mit der steigenden Leistungsfähigkeit von modernen uProcessoren, steigen auch die Anforderungen 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 Echtzeitbetriebssystem (RTOS) bietet einem Entwickler die Möglichkeit ein solches System zu entwerfen. Neben der Echtzeitfähigkeit gibt es aber noch viele weitere Vorzüge für den Einsatz eines Echtzeitbetriebssystems. Durch die Nebenläufigkeit entstehen viele kleine Module, die als separate 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 ist die Problematik des Hardwarenahen Testens. Ein weiterer sehr wichtiger Punkt ist die Entwicklung von energieeffizienten Systemen. In einer Anwendung, die auf einem Echtzeitbetriebssystem 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 detailliert erläutert.

Literatur

1. http://www.acm.org/class/how_to_use.html.
2. M. Y. Ivory and M. A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.

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.

Einhaltung von Prozessdeadline	Komplexität
Skalierbarkeit	Einarbeitungszeit
Erweiterbarkeit	Performance bei aufwändigen Änderungen
Timing Abstraktion	Aufwändige Debugging
Einfache Teamarbeit	
effiziente Energieverwaltung	
3rd Party driver	
Gummi	Bär

Tabelle 1. Vor und Nachteile eines Echtzeitbetriebssystems