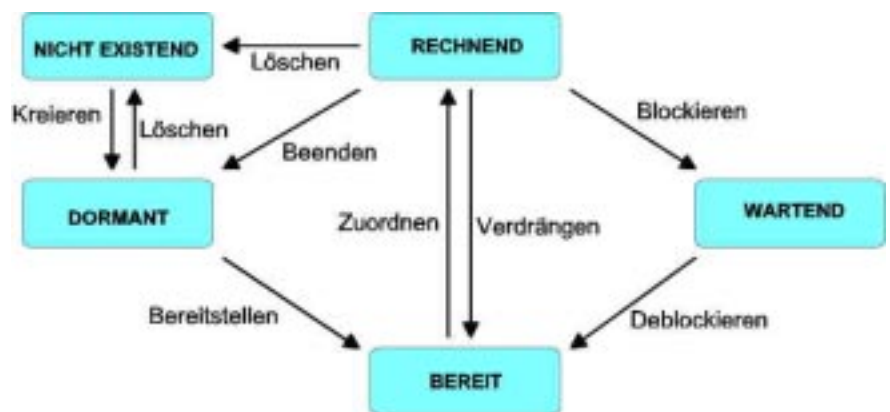


## Echtzeit in der Industrie

Aktuell zum Thema: SICOMP

### Echtzeitbetriebssysteme in der Automatisierung

Der Bereich der industriellen Automatisierung ist ein sehr heterogener Markt. Diese Aussage gilt für den Bereich der Hardware, aber noch mehr für den Bereich der Software. Verglichen mit dem Bereich der PC-Technik gibt es bei Echtzeitbetriebssystemen keinen dominanten Marktführer, vielmehr gibt es viele Hersteller von Echtzeitkernen, Betriebssystemerweiterungen und proprietären Inhauslösungen. Im Vergleich zum PC-Markt sind folglich die Kosten für Systemsoftware relativ hoch.



**Bild 1: Zustandswechsel mit den einzelnen Taskzuständen. Die Zustandsübergänge werden explizit durch Systemaufrufe, externe Ereignisse (Interrupts) oder den Scheduler veranlasst.**

Typische Anforderungen an das Echtzeitverhalten des Rechnersystems stellt z.B. die Automatisierung technischer Prozesse.

- Viele Aufgaben wie digitale Regelalgorithmen, verlangen eine zyklische Bearbeitung, wobei einige Algorithmen sehr empfindlich auf schwankende Zykluszeiten reagieren.
- Für die Bearbeitung von Ablaufsteuerungen ist die Kontrolle von Grenzwerten und Rückmeldungen von asynchronen Ereignissen besonders wichtig.
- Einige Aufgaben müssen zu absoluten Zeiten ausgeführt werden bzw. benötigen absolute Zeiten (z.B. Trend- und Protokoll-Informationen).

Darüber hinaus werden Echtzeitbetriebssysteme zunehmend auch in multimedialen Anwendungen, wie etwa bei der digitalen Audio- und Videotechnik oder im Bereich der Telekommunikation eingesetzt.

### Definition von Echtzeit

In diesem Zusammenhang ist die Definition des Begriffes Echtzeit besonders wichtig. Die Fachliteratur definiert Echtzeit als Forderung nach Gleichzeitigkeit und Rechtzeitigkeit innerhalb quasiparalleler Programmabarbeitung. Nach DIN 44 300 wird Echtzeit als deterministisches Zeitverhalten in einer vorgegebenen Zeitspanne verfügbarer Ereignisse definiert. Eine solche Garantie ist als Standardfunktion (oder Teilfunktion) eines jeden Echtzeitbetriebssystems anzusehen.

In der Praxis wird oft eine Unterscheidung von harter und weicher Echtzeit vorgenommen. Hierbei wird das Überschreiten einer Zeitspanne bei harter Echtzeit als totales Versagen des Gesamtsystems definiert. Bei weicher Echtzeit wird das Überschreiten von Zeitspannen akzeptiert, wobei der entstehende Schaden gering ausfällt.

Die DIN 44 300 reduziert hierbei die Echtzeitfähigkeit auf garantierte maximale Reaktionszeiten (bounded response). Eine weitere wichtige Anforderung an ein Echtzeitbetriebssystem ist aber auch die Mehrprogrammfähigkeit. Die meisten technischen Prozesse lassen sich in mehrere, miteinander gekoppelte und zueinander parallel ablaufende Teilprozesse zerlegen. Dabei ist es äußerst schwierig, vollkommen unübersichtlich und manchmal ganz unmöglich, diese Prozesse durch ein sequentiell ablaufendes Programm zu steuern. Die praktische Realisierung einer Steuerungsaufgabe sieht im wesentlichen so aus, dass eine größere Anzahl von Programmfäden (Task/Thread) verwendet wird, die in einer Monoprozessor-Umgebung quasiparallel ablaufen.

**Tabelle1: Überblick der Echtzeitbetriebssysteme und deren Leistungsdaten (Die Werte für Windows NT 4.0 sind zum Vergleichen mit aufgenommen worden. Alle Angaben sind Herstellerangaben. Vorsicht: Hardware ist nicht immer vergleichbar!)**

	WIN NT 4.0 P/90	WIN CE 3.0 P/90	RMOS 3.20 486/100	VxWorks 68040k/25 Mit Last!	LynxOS 3.0 PII/333	QNX 4 486/100	RTKernel-32 486/33
Interrupt Antwortzeit	10,5us	7,5us	4us	4us	11us	5,6us	5us
Schedulingzeit	?	42,7us	10us	22us	2us	386SX/16 11,1us	5us
Anzahl der Prioritäten	32	256	256	256	256	32	64
Taskanzahl	keine Ang.	keine Ang.	32768	unbegr.	unbegr.	unbegr.	unbegr.
Hartes Ausschalten	nein	HW abhängig	ja	fsck*)	fsck*)	chkfsys*)	ja
Grundtakt	10ms	1ms	1ms, 10ms einstellbar	HW abh. einstellbar	10ms	500us - 50ms einstellbar	HW abh. einstellbar
ROM/FLASH fähig	nein	ja	ja	ja	ja	ja	ja
Speicherschutz	ja	ja	ja	ja	ja	ja	ja

\*) fsck, chkfsys: Programme zum reparieren des Dateisystems.

Beim Design eines Echtzeitbetriebssystems müssen alle Bestandteile sich dem Ziel der Echtzeitanforderung, wie sie oben aufgeführt wurden, unterordnen. Jeder einzelne Treiber muss so implementiert werden, dass er das Gesamtsystem in keinem Fall stört. Besonders die wichtigen und bekannten Blocktreiber wie Festplatten-, Floppy-, Netzwerk-Treiber und TCP/IP-Stack dürfen auch bei großen Datenmengen die CPU-Leistung nur in kleinen Zeitscheiben belegen. Bei guten Echtzeitbetriebssystemen sind die Blockgrößen und Prioritäten der einzelnen Treiber und Dienste einstellbar.

Innerhalb eines Echtzeitbetriebssystems wird nur über die Prioritäten Rechenzeit an die jeweiligen Tasks/Threads vergeben. Im Gegensatz hierzu wird bei Windows ein meldungs-gesteuertes Zuteilungsverfahren verwendet. Ein Thread, der durch seine Start-Priorität einen längeren Zeitraum keine Rechenzeit zugeteilt bekommt, wird künstlich angehoben. Ein im Hintergrund laufender Prozess wird in der Priorität herabgesenkt. Dieses Verhalten darf in einem Echtzeitbetriebssystem nicht vorkommen.

### **Kenngößen von Echtzeitbetriebssystemen**

Für die Anwendung und den Einsatz von Echtzeitbetriebssystemen sind verschiedene Kenngößen von Bedeutung. Hierzu zählen die Interrupt-Antwortzeit, die Kontextwechselzeit und das Verhalten bei periodischen Aufgaben.

Die wohl am häufigsten diskutierte Kenngöße ist die Interrupt-Antwortzeit (interrupt latency). Sie ist die Zeitspanne vom Auslösen der externen Unterbrechung durch die Hardware bis zum Ausführen der ersten Codezeile innerhalb des Interrupthandlers. Dieser Wert wird von allen Herstellern immer ohne Last des Systems gemessen. Ein System im realen Einsatz führt immer Aufgaben durch, die es notwendig machen, innerhalb des Betriebssystems Synchronisierungen durchzuführen. Diese besonderen Codeteile werden durch Abschalten der Interrupts durchgeführt und erzeugen damit eine Vergrößerung der Interrupt-Antwortzeit. Die Kontextwechselzeit (scheduling latency) gibt Auskunft über die Zeitspanne für das Umschalten einer Task. Hierbei werden im Betriebssystem die verschiedenen Register des alten Prozesses gesichert und die Register des neuen Prozesses restauriert.

Beim Verhalten von periodischen Aufgaben ist die Qualität der internen Timer eines Betriebssystems wichtig. Viele Echtzeitanwendungen enthalten Aufgaben die zu einem absoluten oder relativen Zeitpunkt ausgeführt werden müssen. Hierzu stellt jedes Betriebssystem Funktionen zur Verfügung um Tasks zu starten oder neu zu schedulen.

Die Interrupt-Antwortzeiten sind nicht das einzige Maß eines Echtzeitbetriebssystems. Die Qualität eines Betriebssystems kann an weiteren Parametern abgelesen werden.

Für das optimale Echtzeitverhalten einer Anwendung ist es wichtig, jeder Task die richtige Priorität zuzuordnen. Es ist dabei hilfreich vom Betriebssystem viele Prioritäten zur Verfügung zu haben. Moderne Betriebssysteme können 256 Prioritätsebenen verwalten.

Innerhalb eines Echtzeitsystems kann es die Anwendung erfordern, ohne Speichermedium wie Festplatte oder Diskette, das System zu starten. Bei diesen Systemen wird das Betriebssystem von ROM/Flash oder sogar von einem gepufferten RAM gebootet. Hierfür müssen die Betriebssysteme besonders vorbereitet sein.

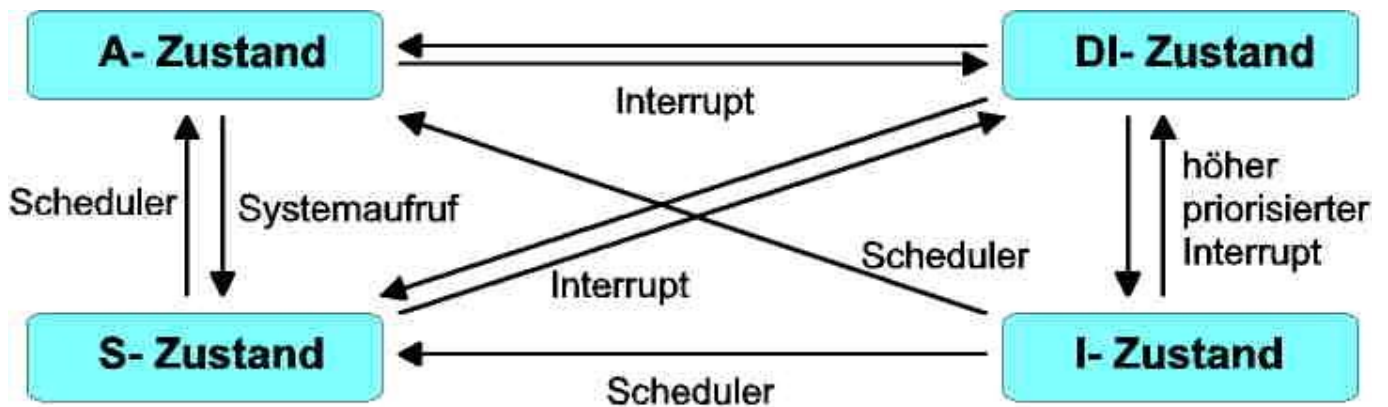
Für einen Softwareentwickler ist es wichtig, einfach und schnell Kontrolle über die jeweilige Hardware zu erlangen. Das Betriebssystem muss es ermöglichen, direkte I/O- und Speicher-Zugriffe zu implementieren. Ebenfalls müssen Interruptroutinen aus einer normalen Anwendung heraus installierbar sein. Ein spezieller Schutz der Hardware, wie er in Windows NT oder Linux vorkommt, erhöht nur die Reaktionszeiten und die Kosten der Entwicklung.

Damit ein System industrietauglich ist, darf ein Spannungsausfall oder ein hartes Ausschalten (Notaus) dem Echtzeitsystem keinen Schaden zufügen. Besonders hiervon betroffen ist das Dateisystem. Ein Schutz des Dateisystems ist mittels einer USV möglich, was aber die Gesamtkosten des Systems erhöht.

Für manche Echtzeitsysteme ist die Anzahl der Interruptquellen bei Standardbetriebssystemen nicht ausreichend. Ein Echtzeitbetriebssystem sollte die Möglichkeit bieten, zusätzliche Interruptquellen einzubinden. Dabei wird eine Kaskadierung von Interruptkontrollern vorge-

nommen. Die Verwaltung der zusätzlichen Interrupts muss dabei vom Betriebssystem mit übernommen werden.

Ein weiterer wichtiger Aspekt der Industrietauglichkeit ist die Unterstützung der Schutzmechanismen der CPU, damit alle Tasks/Threads einer Anwendung untereinander geschützt sind. Daten-, Stack- oder Programmbereichs-Verletzung einer Task/Thread führen zur Fehlererkennung durch das Betriebssystem und zur Ausgabe einer entsprechenden Fehlermeldung, die Hinweise auf Art und Ursache der erkannten Prozessor-Exception gibt.



**Bild 2: Zusammenhang der einzelnen Zustände und ihre Übergänge.**

## RMOS in Automatisierungsaufgaben

Im Folgenden wird das Betriebssystem RMOS (Realtime Multitasking Operating System) in Hinblick auf seine Echtzeiteigenschaften beschrieben. Das Betriebssystem RMOS wird bei der Firma Siemens im Bereich A&D für den Einsatz in Automatisierungsaufgaben vertrieben.

### Multitasking-Mechanismen:

Das Betriebssystem (Scheduler) teilt nur der Task, die bereit zum Rechnen ist und welche die höchste Priorität aller bereiten Tasks besitzt, die CPU zu. Hierbei kann auch eine gerade aktive Task vorzeitig (preemptiv) durch eine gerade bereit gewordene Task höherer Priorität in ihrer Ausführung unterbrochen werden. Da nicht alle Tasks gleichzeitig abgearbeitet werden können, befindet sich jede Task immer in einem von 5 Zuständen:

- RECHNEND: Die Task hat die CPU zugeteilt bekommen, der Programm-Code wird abgearbeitet.
- BEREIT: Die Task ist rechenbereit und wartet auf die Zuteilung der CPU.
- WARTEND: Die Task wartet auf ein Ereignis oder auf einen Zeitpunkt. Bei Eintritt des Ereignisses oder nach Ablauf der Wartezeit wird die Task vom Betriebssystem wieder in den Zustand BEREIT gesetzt.
- DORMANT: Die Task ist im Ruhezustand. Ihre Existenz ist innerhalb des Betriebssystems bekannt. Sie kann durch eine andere Task gestartet werden.
- NICHT EXISTENT: Die Existenz der Task ist innerhalb des Betriebssystems nicht bekannt. Der Programm-Code der Task kann entweder im Arbeitsspeicher sein oder von einem Massenspeicher nachgeladen werden.

## Koordination und Kommunikation zwischen Tasks

Für die Task-Verwaltung bietet RMOS im wesentlichen folgende Dienste:

- Task kreieren, löschen, nachladen, starten, beenden und zyklischer Taskstart.
- Gezieltes Aufheben der Pause einer anderen Task.
- Gezieltes Anhalten bzw. Freigeben des Schedulers.
- Starten einer Task bei erkennen eines Hardware- oder Software-Interrupts.
- Dynamische Prioritätsänderung (eigene oder andere Task).
- Statusabfrage (eigene oder andere Task).

Die Task-Kommunikation wird bei RMOS durch den lokalen Botschaftenverkehr und Ereignisflags ermöglicht. Der Botschaftenverkehr basiert auf Mailboxen, zu denen rechnende Tasks Nachrichten senden bzw. von diesen empfangen können. Dabei können die Tasks auswählen, ob sie beim Senden auf das Abholen ihrer Nachricht oder beim Empfangen auf das Eintreffen einer Nachricht warten wollen.

Die Koordination des wechselseitigen Ausschlusses von Tasks bei gleichzeitigen Zugriff auf gemeinsam benutzte Betriebsmittel (Datenstrukturen, Ein/Ausgabebausteine, ...) wird mit Hilfe von Semaphor-Aufrufen gesteuert.

Eine Task kann sich über Ereignisflags mit anderen Task synchronisieren und Zustände anzeigen. Typischerweise wird ein Ereignisflag durch eine Task gesetzt, wenn eine Bedingung erfüllt oder durch einen Systemprozeß beeinflusst wurde, wenn ein Ereignis (z.B. Interrupt) eingetroffen ist. Ein Flag entspricht dabei einer binären Nachricht mit vordefinierter Bedeutung, die durch die Applikation festgelegt wird. Flags bieten eine eleganten und effiziente Form für den Austausch von binären Nachrichten.

## Interruptbehandlung:

Die Interrupt-Service-Routinen werden direkt in einer Hochsprache (z.B. in C) vom Anwender codiert und können durch Prozedur-Deklaration problemlos dynamisch installiert, aktiviert oder ausgetauscht werden. Das Schreiben eines speziellen RMOS-konformen Treibers zur Behandlung der Echtzeit-Interrupts ist damit in der Regel nicht notwendig.

Die Interrupt-Behandlung unter RMOS ist in vier Betriebszustände unterteilt:

- DI-Zustand, Disable Interrupt: alle Interrupts sind gesperrt.
- I-Zustand, Interrupt: Interrupts, die eine niedrigere Priorität als der momentan abgearbeitete haben, sind gesperrt. Systemaufrufe sind erlaubt aber sie werden erst nach Beendigung des I-Zustandes ausgeführt. D.h. das Ergebnis eines Systemaufrufes steht nicht zur Verfügung. Es sollten demnach nur Systemaufrufe verwendet werden, die dem Betriebssystem etwas mitteilen, wie Task starten, Flag setzen, Semaphore zurücksetzen.
- S-Zustand, System: alle Interrupts sind freigegeben und das Absetzen nicht blockierender Betriebssystemaufrufe ist möglich. Der Wechsel in den A-Zustand ist erst nach Beendigung aller Aktivitäten im S-Zustand möglich.
- A-Zustand, Applikation: alle Interrupts sind freigegeben, Betriebssystemaufrufe können beliebig abgesetzt werden.

Bei einem Ausblick auf zukünftige Entwicklungen im Bereich von Echtzeitsystemen wird JAVA zu einer ernstzunehmenden Alternative. JAVA hat innerhalb von nur drei Jahren die Informationstechnik revolutioniert. Die neue Programmiersprache wurde vom Spielzeug für blinkende Webseiten zur etablierten Automatisierungsplattform. Mit Verbreitung der Sprache JAVA ist das Betriebssystem nicht mehr ausschließliches Kriterium des Gesamtsystems. JAVA als hardware- und betriebssystemunabhängige Sprache ermöglicht es ebenfalls

Echtzeitanwendungen zu entwickeln. Dabei treten die Eigenschaften des unterlagerten Betriebssystems mehr und mehr in den Hintergrund.

Zum Zweck einer Standardisierung für Echtzeitanwendungen in JAVA hat sich seit geraumer Zeit das J-Consortium gebildet. Das Ziel ist es hierbei eine Echtzeiterweiterung für JAVA zu definieren und als Standard zu etablieren. Es liegt bereits eine Draft-Version vor. (<http://www.j-consortium.org>).

Eine Verabschiedung der Draft ist für Juni 2000 vorgesehen.

In diesem Zusammenhang lassen sich die beiden SIEMENS Produkte Sicomp RTVM (**R**eal **T**ime **V**irtual **M**achine) und Sicomp JFPC (**J**ava **f**or **P**rocess **C**ontrol) erwähnen, die unter RMOS echtzeitfähig sind (<http://www.ad.siemens.de/sicomp>) (vgl. auch SPS-Magazin Ausgabe: 1+2/2000, S. 120ff.).

Autor:  
Dipl.-Ing. Mirko Janetschke  
E-mail: [mirko.janetschke@fthw.siemens.de](mailto:mirko.janetschke@fthw.siemens.de)

Zeitschrift: SPS-Magazin  
Ausgabe: 6+7/2000

Weitere Informationen unter: <http://www.ad.siemens.de/sicomp>  
Electronic Commerce: <http://mall.siemens.de>

Siemens AG  
Automatisierungs- und Antriebstechnik  
Systems Engineering, A&D SE  
Postfach 23 55, D-90713 Fürth  
E-mail: [sicomp@fthw.siemens.de](mailto:sicomp@fthw.siemens.de)

Änderungen vorbehalten

Siemens Aktiengesellschaft

