Realzeit-Betriebssysteme für Mikrocomputersysteme

Helmut Reichel

Rohlfsstr. 15

8000 München 81

Leistungsfähige Elektroniksysteme auf der Basis von Mikroprozessoren sind heute integraler Bestandteil der unterschiedlichsten Steuerungssysteme, vom einachsigen Elektromotorantrieb bis zum programmierbaren Industrie-Roboter. Alle diese Anwendungen haben einen gemeinsamen Kern: zeitlich parallel laufende Aktionen müssen aufeinander abgestimmt wohlgeordnet abgewickelt werden. Standardisierte Betriebssysteme stellen der Anwendungsebene diesen Verwaltungs- und Dienstleistungsanteil zur Verfügung. Verbunden damit ist die Unterstützung für Entwicklung, Test und Pflege der Anwendungssoftware.

Diese Themen werden im folgenden weiter erörtert:

- Grundsätzliche Merkmale von Realzeitbetriebssystemen
  Parallelität, Synchronisation, Tasking, Verwaltung der Resourcen,
- Abgrenzung zu Multi-user-Systemen
- Reaktionszeiten
- Software-Entwicklung und Portabilität von Anwendungssoftware
- Trends

1. Grundsätzliches zu Realzeit-Betriebssystem

Günstiger Preis und hohe Flexibilität der Mikroprozessortechnologie ermöglichen es, Steuerintelligenz zum integralen Bestandteil einfacher und komplexer Anlagen zu machen.
Realzeitsysteme überwachen und steuern Schaltanlagen, Fertigungsmaschinen, Roboter, Turbinen, Telefonanlagen, Laborgeräte, Waschmaschinen, etc., vom einzelnen Schrittmotor bis zur Produktionsanlage. Typisch ist die Ausstattung mit Sensoren und Effektoren anstelle der klassischen Peripheriegeräte der Büro-EDV.

Das entscheidende Merkmal jedes RT-OS (Real Time Operating System) ist seine Fähigkeit, Meldungen der Prozeßperipherie über asynchron eintretende Ereignisse entgegenzunehmen und auf sie "sofort" zu reagieren. Voraussetzungen dazu sind:

- Das Hardware/Software-System stellt einen Eingang für ein externes Signal zu Verfügung. Im allgemeinen löst das externe Signal einen Hardware-Interrupt aus, der zum Wechsel des Betriebszustandes des Rechners führt.
- Der Rechner ist in der Lage, mehrere konkurrierende Funktionen quasi parallel abzuwickeln. Damit unterscheidet er sich vom einfachen Personal-Computer, der seine Aufgaben streng sequentiell abarbeitet.
- Die Reaktionszeit des Rechners bleibt auch bei der ungünstigsten Folge der externen Ereignisse unter einer für diese Anwendung akzeptablen Grenze. Diese Vorhersagbarkeit der begrenzten Reaktionszeit auch unter starker Systembelastung unterscheidet ein Realzeit-System vom Multi-user-System.

Die Koordination parallel ablaufender Vorgänge ist die allen Anwendungen gemeinsame wichtigste Leistung des RT-OS. Es stellt die Werkzeuge dazu der Anwendungsebene zur Verfügung. Darüber hinaus macht das Betriebssystem die Anwendungssoftware unabhängiger von der unterlegten Hardware, es erzeugt Portabilität von Anwendungssoftware.
Die konzeptionellen Schwerpunkte und Unterschiede der miteinander verwendeteten Betriebssystemarten zeigt die folgende Aufstellung:

Typische Unterschiede zwischen

Realzeit-Betriebssystem	**Multi-user-Betriebssystem**
- Anwendung: Maschinensteuerung	- Anwendung: Mehrplatzsystem
- Hoher Systemdurchsatz und begrenzte Reaktionszeit	- Gleichmäßige Auslastung der System-Resourcen
- schnelle Interruptabarbeitung	- Zuteilung von Rechenleistung über Zeitscheiben
- kurze Interruptsperrzeiten	- Zwischenpufferung von Daten
- Betriebssystemkern kann in die Anwendungssoftware integriert sein	- strikte Trennung zwischen Betriebssystem und verschiedenen Anwendungspaketen
- Mehrere Prozesse innerhalb einer geschlossenen Anwendung	- Mehrere Anwendungen in mehreren Prozessen

2. Funktionen von Realzeit-Betriebssystemen

2.1 Prozeß- bzw. Task-Verwaltung

Ein Anwendungsprogramm zur Lösung einer Teilaufgabe läuft als ein Prozeß bzw. Task unter Systemverwaltung ab. Ein Prozeß ist gekennzeichnet durch

- seinen Namen bzw. seine Identifikationsnummer
- seinen Zustand (rechnend - bereit - wartend)
- seine Priorität gegenüber anderen Prozessen
- seine Betriebsmittel (Speicherplatz, Stackbereich, Ein-/Ausgabekanäle)

Das Betriebssystem leistet die nur statische oder auch dynamische Erzeugung eines Prozesses. Weitere Systemaufrufe regeln die Prozeßpriorität und die statische oder (seltener unterstützt) auch die dynamische Zuteilung von Speicher, Stack- und Ein-/Ausgabebereichen.

Die Strategie bei der Regelung des Prozeßzustandes ist der sensitivste Teil der Prozeßverwaltung. Die gleichmäßige, von der Priorität gesteuerte Zuteilung von Rechenzeit zu den rechenwilligen Prozessen und die Strategie bei der Überprüfung, welche Prozesse zu welchem Zeitpunkt in den Wartezustand versetzt oder reaktiviert werden können, bestimmen die Reaktionszeiten wesentlich mit. Die Prozeßwechselzeiten liegen dabei zwischen 0.1 und 2.0 ms.

2.2 Kommunikation und Synchronisation

Nachrichten werden zwischen Prozessen übermittelt, wenn Ereignisse oder Daten in andere Abläufe hineinspielen. Übliche Methoden sind "Mailboxes" (Briefkasten) und "Queues" (Nachrichten-Warteschlangen). Das RT-OS stellt Zwischenpuffer für Nachrichten zur Verfügung und entkoppelt so die Prozesse voneinander.

Eine Mailbox ist schneller zu bedienen und einfacher zu handhaben, sie kann jedoch nur eine 1:1-Verbindung herstellen, die nur eine Nachricht enthält. Queues sind dann notwendig, wenn ein sendender Prozeß schneller Nachrichten absendet als sie der Empfänger abholen kann. Außerdem kann über Queues jeder Prozeß mit jedem anderen Daten austauschen. Im RT-OS ist hier jedoch wieder sicherzustellen, daß jeweils der Empfänger mit der höchsten Priorität seine Nachricht vorrangig erhält, um wichtige Ereignisse schnellstens bearbeiten zu können.
Es können Prozeßzustandswechsel mit der Kommunikation verbunden sein, z.B. wenn ein Prozeß erst nach Erhalt der Nachricht weiterarbeiten kann.
Der Zeitbedarf für den Austausch von Nachrichten liegt zwischen 0.05 und 1.0 ms.
Einige RT-OS verfügen zusätzlich zur Beschleunigung der Prozeßsynchronisation über "Semaphore". Dies sind Zählvariablen, die dann eingesetzt werden, wenn keine Daten zu übermitteln sind, sondern nur das Ende eines Ablaufes einen anderen Prozeß mitgeteilt werden soll. Auch Speicherbereiche können für mehrere Preozesse als zugänglich erklärt werden, um den Zugriff auf globale Daten zu ermöglichen.

2.3 Interruptauswertung und Realzeituhr

Interrupt-Serviceroutinen stellen die Verbindung zwischen externen Ereignissen und den Bearbeitungsprozessen her. Diese Serviceroutinen sind, i.a. geräte- bzw. anwendungsspezifisch zu lösen, die Betriebssystemhersteller liefern jedoch meist typische Musterprogramme in Assemblercode. Das RT-OS übergibt bei einem Interrupt die Regie der jeweiligen Serviceroutine, welche die erste Aufbereitung übernimmt und Nachrichten dem Betriebssystem zur Weiterleitung an die Anwendungsprozesse übergibt.

Beim Entwurf von Serviceroutinen ist zu beachten, daß Sperrzeit für andere Interrupts möglichst kurz bleiben, um beim Auftreten einer Häufung von Interrupts reaktionsfähig zu bleiben.

Eine spezielle Quelle von Interrupts ist die Realzeituhr, die in Hardware realisiert ist, und deren hohe Auflösung im ms-Bereich den zeitgenauen Start von Aktivitäten ermöglicht. Das RT-OS stellt Systemaufrufe bereit, durch die sich Prozesse für eine frei programmierbare Zeit in Wartezustand versetzen können.

3. Implementierungen von Realzeit-Betriebssystemen

Einige Vertreter von RT-OS sind (ohne Anspruch auf Vollständigkeit und ohne Wertung) in der folgenden Tabelle aufgeführt:

Name	Hersteller	Hardwarebasis
VRTX	Hunter & Ready	Z8002, M68000, i8086, i8080
OS-9	Microware	68000, M6809
iRMX	Intel	Intel-CPU's
VERSAdos	Motorola	Motorola-CPU's
RTIX	Kontron	i8086, M68000

Allen ist gemeinsam, daß zu einem festen Kern für die Applikation benötigte Standarderweiterungen und anwendungsspezifische Treiber hinzugefügt werden können. ROM-fähige Implementierungen sind möglich, ebenso Erweiterungen zu dem Anschluß von Massenspeichern wie Platten, Floppy-Disk bzw. Massenspeicher. Die von diesem System unterstützte Hardwarebasis beginnt bei 4 KB ROM und reicht bis zum PC-ähnlichen System.

4. Software-Entwicklung für Realzeitsystem

Mehr und mehr setzt sich die Sprache "C" zur Programmierung von Realzeitsystemen durch. Meist wird die Anwendung auf speziellen Entwicklungssystemen erzeugt und auf Funktionalität vorgetestet. Die modulare Struktur erleichtert die Programmerstellung unter Verwendung von Modulen aus Rahmensoftware und Bibliotheken.
Für den Test auf der Zielhardware stehen generell Debug-Programme zur Verfügung an deren Komfort hohe Ansprüche gestellt werden.

Die Portabilität der Anwendungssoftware wird durch die Verwendung der auf allen Rechnern verfügbaren Sprache "C" und die unterliegenden Betriebssystemschichten zumindest innerhalb einer Betriebssystemfamilie gewährleistet.
Wichtig für die Betriebssicherheit in komplexen Anwendungen sind Hardware-unterstützte Überwachungen auf z.B. Speicherzugriffe auf außerhalb des aktiven Prozesses liegende Daten.

5. Trends

Zwei ganz unterschiedliche Trends zeichnen sich derzeit ab: zum einen der erhöhte Hardware-Einsatz in Multiporzessorsystemen und in Netzwerken, zum anderen die Bestrebungen, auf zunächst nicht RT-fähige Systeme wie Unix oder CP/M einen RT-Kernel aufzusetzen.

Der erste Weg ist bei "Super-Mikros" bereits häufig begangen: ein Subsystem mit eigener CPU und Speicher steuert unter einem RT-OS die Peripherie, ein weiteres Subsystem bedient eine zeitkritische DFÜ-Leitung, etc. Dieser Hardwareeinsatz reduziert die Zahl der Interrupts auf der Haupt-CPU und sorgt so für die Steigerung des Durchsatzes. Entsprechendes leistet der Rechnerverbund über lokale Netzwerke.

Von den Multi-user-Anlagen her entwickelt sich der andere Trend: man setzt auf die dort etablierte Betriebssoftware (UNIX, CP/M, MS-DOS) einen RT-Kernel, der die Interruptverarbeitung optimiert und sich seinerseits für weitergehende Dienstleistungen wie z.B. der Plattenverwaltung des unterliegenden Betriebssystems bedient. Für gemischte Anwendungen kann dies ein akzeptabler Lösungsweg sein.

Insgesamt sorgen 16- und 32-Bit-CPU's auch im RT-Sektor für eine wesentliche Erhöhung des Leistungspotentials bei unterproportional steigenden Kosten: damit ist die Möglichkeit offen, mehr Komfort und höhere Funktionsdichte untereinander zu verbinden.