Harte Echtzeit unter Windows XP, Windows 2000 und XP Embedded

Echtzeiterweiterungen sorgen für Determinismus und sicheren Betrieb

Anwendungen in der Mess-, Steuer- und Regeltechnik erfordern vom Betriebssystem häufig hohe Datentransferraten und kurzen Reaktionszeiten und setzen gleichzeitig vermehrt die Unterstützung von Visualisierungs- und Kommunikationsfunktionen voraus. Die Kombination aus Standardbetriebssystem und Echtzeiterweiterungen können diese Anforderungen erfüllen.

JÜRGEN RALL

Jürgen Rall ist ??? Sybera

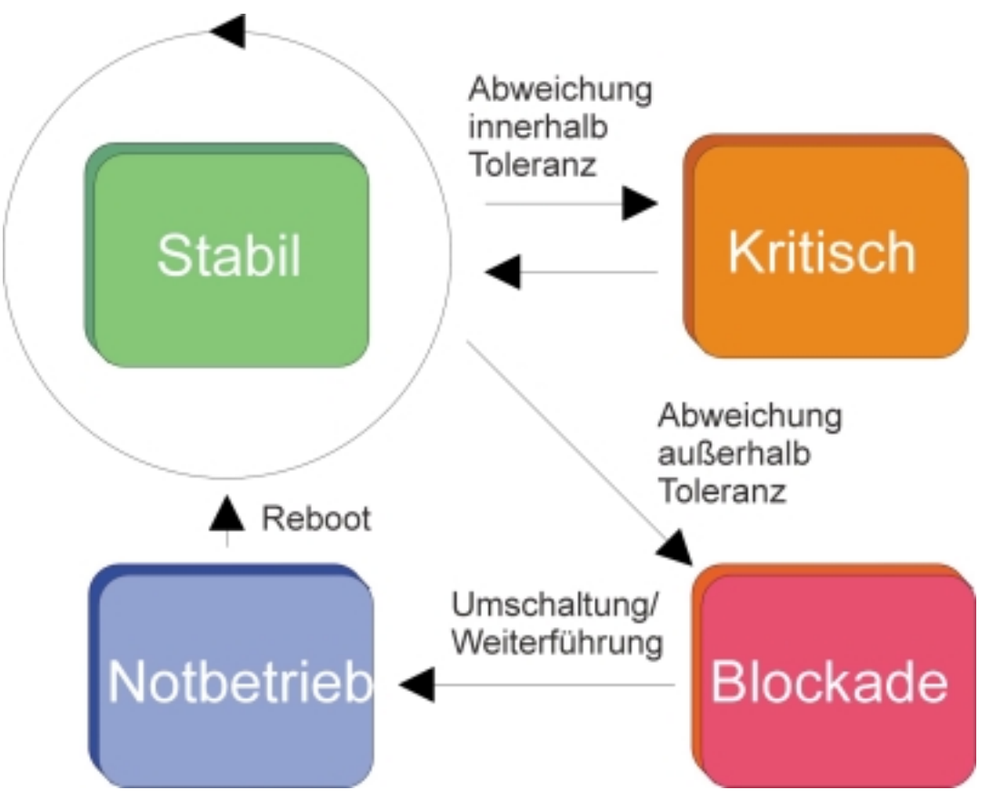

Abb. 1: Erforderliche Betriebszustände in einem Echtzeitbetriebssystem

Echtzeitbetriebssysteme werden immer dann benötigt, wenn zeitkritische dynamische Systeme zuverlässig verwaltet werden müssen. Die Verletzung des deterministischen Zeitverhaltens führt bei sogenannten sensitiven Systemen i.d.R. zu einem kritischen bis katastrophalen Zustand (z.B. Linearsteuerung, Füllmengenüberwachung), während nicht-sensitive Systeme (z.B. Geschwindigkeitsregelung) bei sporadischer Verletzung temporär in einen kritischen Zustand wechseln oder stabil bleiben.

Mussten in den Anfängen Echtzeitbetriebssysteme in erster Linie die Anforderungen ‚deterministisches Zeitverhalten' und ‚Sicherheit' erfüllen, müssen sie heute vermehrt zusätzlich standardmäßig Visualisierungs- und Kommunikationsfunktionen unterstützen. Dazu gehört die mehr oder weniger vollständige Unterstützung der gesamten Palette der Kommunika-

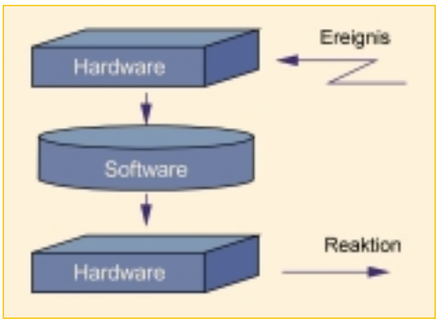

Abb. 2: Die Gesamtlatenzzeit zwischen Ereignis und Reaktion ergibt sich aus der Summe der Einzellatenzzeiten eines Systems

tionsschnittstellen, wobei jedoch zwischen echtzeitfähiger und nicht-echtzeitfähiger Kommunikation zu unterscheiden ist. Echtzeitfähige Kommunikation wird heute vor allem bei CAN-, Ethernet- und serieller Verbindung gefordert wenn der Datenfluss entweder nicht unterbrochen werden darf oder Datenpakete in einem festgelegten Zeitraster bearbeitet werden müssen.

Vor allem in der industriellen Steuerungstechnik ist die grafische Anzeige der Daten zur Interaktion zwischen Mensch und Maschine eine wichtige Anwendung. Moderne Echtzeitbetriebssysteme bieten daher i.d.R. eine eigene grafische Schnittstelle zur Datenvisualisierung. Um den Echtzeitablauf nicht zur beeinflussen, ist die Visualisierung jedoch von der Datenaufnahme zeitlich entkoppelt. Hierbei ist die Programmierbarkeit des Echtzeitbetriebssystems ein wichtiger Faktor für die Erstellung von Applikationen und deren Wartung und dem damit verbundenen Kostenfaktor. Ein modernes Echtzeitbetriebssystem muss darüber hinaus einen Notbetrieb gewährleisten, um z.B. im Fehlerfall einen Roboter aus einem Gefahrenbereich manövrieren zu können. Besonders bei der Steuerung sensitiver Systeme muss der Notbetrieb in der Lage sein, die Gesamtsystemstabilität aufrecht zu erhalten.

Definition

Eine weitverbreitete Unsicherheit besteht zudem bei den Begriffen harte und weiche Echtzeit. Aus der Literatur lässt sich folgende Definition herleiten:

- Bei Betriebssystemen, die die Anforderung harte Echtzeit erfüllen sollen, muss die Latenzzeit der Echtzeit-Tasks unabhängig von dem Verhalten anderer Tasks sein.
- Bei weicher Echtzeit kann die Latenzzeit der Echtzeit-Task abhängig von dem Verhalten anderer Tasks sein, jedoch nur innerhalb einer vorgegebenen Abweichung.

Des weiteren muss bei der begrifflichen Definition zwischen deterministischem Zeitverhalten von Aktio und Reaktio unterschieden werden, sofern eine periodische Programmabarbeitung zugrunde liegt.

- Bei der umlaufenden Echtzeit (Aktio) wird eine Echtzeit-Task periodisch aufgerufen, welche die Steuerung ausführt. Die Umlauf-Periode bestimmt hierbei das deterministische Zeitraster und somit die Genauigkeit.
- Bei der ereignisgesteuerten Echtzeit (Reaktio) wird die Echtzeit-Task immer dann aufgerufen, wenn das Ereignis eintritt (z.B. periodisches Ereignis). Die Latenzzeit muss hierbei kleiner als die Periodendauer des Ereignisses sein (Voraussetzung: Latched Signal). Die Latenzzeit bis zur Abarbeitung des Ereignisses ist hierbei von Software und Hardware abhängig (z.B. Prioritätenverteilung der Interrupts).

Die Gesamt-Latenzzeit zwischen Ereignis und Reaktion ergibt sich aus der Summe der Einzel-Latenzzeiten eines Systems.

Echtzeit-Subsystem

Durch die Kombination eines nicht-echtzeitfähigen Betriebssystems mit einem s.g. echtzeitfähigen Subsystem können die Vorteile beider Systeme vereint werden. So können die Visualisierungs- und Kommunikations-Eigenschaften des nicht-echtzeitfähigen Betriebssystems mit dem deterministischen Zeitverhalten des echtzeitfähigen Subsystems für Steuerungsaufgaben ideal genutzt werden.

Ein deterministisches Zeitverhalten ist unter den Betriebssystemen Windows (NT, 2000 oder XP) nur mit Echtzeit-Subsystemen realisierbar. Bei reinen Softwarelösungen muss bei solchen Systemen aufgrund des Betriebsmanagements allerdings meist mit erheblichen Abweichungen (Jitter) gerechnet werden.

Sybera hat daher unter der Bezeichnung SHA (Sybera Hardware Access) eine Echtzeit-Engine

Abb. 3: Modelldarstellung der Zusammenarbeit von Betriebssystem, Echtzeiterweiterung und Anwendung

Abb. 4: Das Echtzeitsubsystem Sybera Hardware Access

entwickelt, die eine vollständige Abkopplung der Echtzeit-Task zum bestehenden Betriebssystem umsetzt und somit geringstes Jitter-Verhalten aufweist. Während bei herkömmlichen Echtzeit-Subsystemen die Realtime-Task bei zunehmender CPU-Belastung starken Zeitschwankungen unterliegt, bleibt die Realtime-Task von SHA auch bei extremer CPU-Belastung zeitstabil. Somit kann ein genaues Echtzeit-Verhalten für industrielle Anwendungen unter Windows (NT, 2000, XP, XP Embedded) ohne zusätzliche Hardware umgesetzt werden.

Mit Hilfe der SHA-X-Realtime-Engine sind Echtzeit-Task-Zyklen bis zu 10 ms realisierbar (100 KHz Abtastrate). Ein integriertes Watchdog-System überwacht die Echtzeit-Tasks und ermittelt die verbleibende Task-Zeit. Das SHA-X-Failsafe-System bietet zusätzlich die Möglichkeit, auch bei schweren Ausnahme-Fehlern (z.B. Blue-Screen) das System mit einer Rescue-Task aktiv zu halten oder kontrolliert zu beenden. Mit dem Failsafe-System kann beispielsweise ein Roboterarm aus der Gefahrenzone herausgefahren und ein Alarmsignal ausgelöst werden.

Der Entwickler arbeitet mit dem SHA System innerhalb seiner gewohnten Entwicklungsumgebung (z.B. Visual C++). Zusätzlich ermöglicht SHA den Zugriff auf alle Hardware-Ressourcen direkt von der Applikationsebene. Ob I/O-Port-Zugriffe, Mapped Memory, Timer- und Interrupt-Steuerung oder Echtzeit, alles kann ohne Zeitverlust und ohne aufwendige Gerätetreiber-Programmierung realisiert werden.

Zu den unterstützten Plattformen zählt ab der Version SHA 7Plus auch Windows CE. Hierbei wurde dasselbe Interface implementiert wie bei der Windows NT-, 2000- und XP-Version. Ein unter Windows NT, 2000 oder XP geschriebenes Programm kann somit leicht nach CE portiert werden. Aktuell werden die Prozessoren x86, ARM und SH4 unterstützt, weitere Plattformunterstützungen sind in Planung.