Zeitmessungen an Echtzeitbetriebssystemen

A.Riegg,V. Scheub

Universität Stuttgart
Institut für Regelungstechnik
und Prozeßautomatisierung

1. Einleitung

Bis vor kurzem wurden Personal-Computer hauptsächlich im Büro-Bereich für ´off-line´ Aufgaben eingesetzt, während für technische Anwendungen überwiegend in Assembler programmierte spezielle Mikrorechner zum Einsatz kamen. Inzwischen geht der Trend dahin, Standard-PCs mit entsprechender Peripherie zu versehen und als ´Mini-Prozessrechner´ in direkter Kopplung mit dem technischen Prozess einzusetzen. Dieser Trend wird unterstützt durch die auf diesen Rechnern angebotenen höheren Programmiersprachen.

Die Verwendung von Standard-Bussystemen ermöglichte ein großes Angebot von Peripherie-Karten für die unterschiedlichsten Anwendungen. Entscheidend ist aber die Tatsache, daß in letzter Zeit zunehmend Echtzeit- und Multitasking-Betriebssysteme angeboten werden, die die Programmierung von Echtzeitaufgaben durch Systemfunktionen unterstützen und vereinfachen.Die Verwendung höherer Programmiersprachen wird ermöglicht, was zu einer Verkürzung der Programmentwicklungszeiten führt.

Für den PC-Anwender eröffnen sich neue Anwendungsmöglichkeiten, es ergibt sich aber auch das Problem, einerseits die neuen Programmkonzepte wie Multitasking, Scheduling, Synchronisierung etc. zu erlernen und andererseits die angebotenen Betriebssysteme zu beurteilen und zu vergleichen. Der vorliegende Beitrag betrachtet einen wesentlichen Aspekt der Beurteilung, nämlich die Messung von Ausführungs-und Reaktionszeiten von Echtzeitsystemen.

2. Was ist ein Echtzeitsystem

Eine Definition von Echtzeitsystemen, die den Sachverhalt in wenigen Worten beschreibt, ist folgende:/1/

> "Echtzeitsysteme sind Systeme, die mit Ihrer Umgebung unter harten zeitlichen Bedingungen kommunizieren, wobei das verwendete Rechensystem in einem geschlossenen Wirkungskreis steht."

Der etwas vage Ausdruck ´harte zeitliche Bedingungen´ kann folgendermaßen präzisiert werden:

> "Die Zeitbedingungen sind hart, wenn von ihrer Einhaltung das korrekte Arbeiten des Rechensystems im Echtzeitbetrieb abhängt."

Anschaulich bedeutet dies, daß es für das korrekte Arbeiten eines Realzeitsystems erforderlich ist, daß z.B. auf das Auftreten eines Signals vom technischen Prozess innerhalb fest vorgegebener zeitlicher Grenzen eine entsprechende Reaktion erfolgen muß, z.B. das sofortige Abschalten eines Gerätes bei Überdruck.

Die vorgegebenen zeitlichen Grenzen hängen ihrerseits fast ausschließlich von der zu realisierenden Aufgabe ab. So ist die Steuerung und Regelung eines Industrieroboters nur mit sehr kurzen Abtast- und Reaktionszeiten des Steuerrechners machbar, während die Steuerung einer Heizungs- und Klimaanlage bezüglich der zeitlichen Anforderungen relativ unkritisch ist.

Zum Betrieb eines Echtzeitsystems werden einige charakteristische Funktionseinheiten, realisiert durch verschiedene Komponenten, benötigt, die es von anderen Rechensystemen unterscheiden. Es sind dies:

1. Der E/A-Verkehr: er bewerkstelligt den direkten Datenaustausch mit dem technischen Prozess und wird durch die Peripherie-Karten und Treiberprogramme abgewickelt.
2. Die Quasiparallele Auftragsabwicklung bei Einprozessor-Rechnern: sie dient zur Organisation des Multitasking und wird durch das Betriebssystem realisiert.
3. Die Korrekte Synchronisierung parallel ablaufender Programme: sie ist Aufgabe des Betriebssystems und einer korrekten Anwenderprogrammierung.
4. Rechnerische Behandlung von Zeitbedingungen: zur zeitabhängigen Steuerung des Programmablaufs wird realisiert durch eine Echtzeituhr und entsprechende Möglichkeiten der Programmiersprache.

Die Struktur eines solchen Systems und das Zusammenspiel der dafür notwendigen Komponenten ist in Bild 1 veranschaulicht.

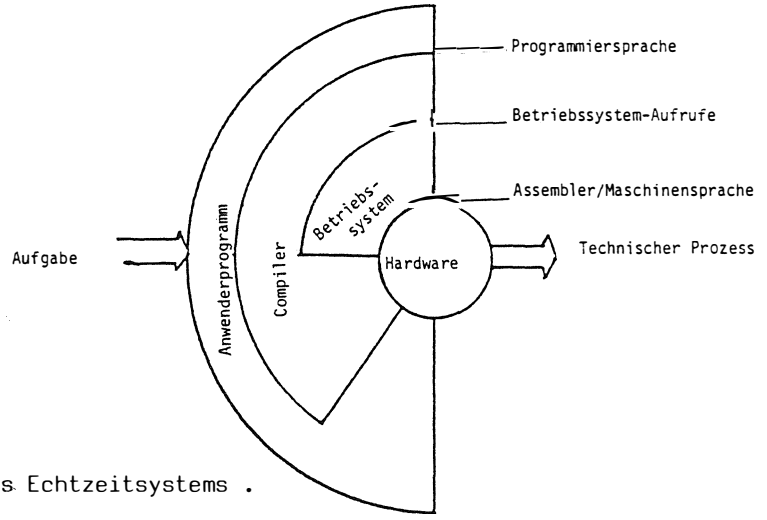

Bild 1: Struktur eines Echtzeitsystems .

Die verschiedenen Komponenten sind durch wohldefinierte Schnittstellen wie Maschinensprache, Betriebssystemaufrufe, Programmiersprache und Spezifikation des Anwenderprogramms getrennt. Die Leistungsfähigkeit dieser Komponenten und der Schnittstellen bestimmt die Leistungsfähigkeit des Gesamtsystems:

1. Genauigkeit/Korrektheit der Aufgabenstellung
2. Programmierstil/Programmierumgebung
3. Eignung der Programmiersprache
4. Effizienz des Compilers
5. Effizienz des Betriebssystems
6. Effizienz der Hardware

Während zur Beurteilung der Leistungsfähigkeit eines Systems Aufgabenstellung und Programmierstil nicht objektiv meßbar sind, lassen sich Effizienz von Compiler, Betriebssystem und Hardware durch geeignete Verfahren feststellen. Ein wesentlicher Punkt bei der Beurteilung der Effizienz von Echtzeitsystemen ist das Zeitverhalten. Zur Messung des Zeitverhaltens im Sinne von verbrauchter CPU-Zeit wird im folgenden das in DIN 19 242 vorgestellte Verfahren beschrieben, das hierfür besonders gut geeignet ist.

3. Leistungstest von Prozeßrechensystemen - Zeitmessungen nach DIN 19 242

Die in dieser Norm beschriebenen Tests dienen dazu, das Zeitverhalten von Prozeßrechensystemen zu ermitteln, um die Eignung eines solchen Systems für eine vorgegebene Aufgabe beurteilen zu können.

Voraussetzung ist eine genau Analyse der Aufgabenstellung und Bestimmung der erforderlichen zeitlichen Randbedingungen, um die erhaltenen Messergebnisse beurteilen zu können. Je nach Aufgabenstellung kann der Schwerpunkt der Untersuchung auf unterschiedlichen Kriterien beruhen:

- Ausführungszeit kritischer Rechenoperationen wie Arithmetik, Unterprogrammaufrufe etc.
- Interrupt-Reaktionszeit
- Taskwechsel
- E/A-Vorgänge
- Anwender-Komfort

Die mit den Tests gemachten Messergebnisse sind dann entsprechend zu gewichten (Anwendungsprofil).

3.1 Das Testverfahren nach DIN 19 242

Als Testergebnisse einer solchen Testreihe erhält man die Laufzeiten bestimmter Anweisungsfolgen. Der Wichtigste Grundgedanke des Normentwurfs ist, die Zeitdauern für ganz bestimmte Aufgaben, genannt Testfunktionen, zu erhalten, wobei die einzelnen Testfunktionen unabhängig von einer verwendeten Programmiersprache definiert sind. Als Zeit dauern werden hierbei sowohl die Verweilzeiten, also die Zeit, die ein Programmstück effektiv benötigt, als auch die Zentralprozessorzeit, die es beansprucht gemessen. Die Zeiten sind verschieden, wenn in einem Programm E/A-Wartezeiten o.ä. mit Taskwechsel vorkommen.

Bei der Messung der Zentralprozessorzeit, die eine bestimmte Anweisungsfolge benötigt, bedient sich das Testverfahren einer indirekten Messmethode. Hierzu werden mindestens zwei Programme verwendet. Das eine heißt Hintergrundprogramm und wird mit niederer Priorität als erstes gestartet. Das zweite Programm, das die eigentliche Testfunktion enthält, wird als Vordergrundprogramm mit höherer Priorität anschließend gestartet. Die Laufzeit des Hintergrundprogrammes verlängert sich somit immer genau um die Zeit, die das Vordergrundprogramm den Zentralprozessor belegt. Daß es sich genau um die Zentralprozessorzeit des Vordergrundprogramms handelt, begründet sich darin, daß als Hintergrundprogramm eine rein arithmetische Anweisungsfolge genommen wird, die keine E/A-Wartezeiten benötigt. Da natürlich jedes Testprogramm nicht nur aus den interessierenden Anweisungen bestehen kann, sondern zwecks Bedienung, Durchführung und Protokollierung noch weitere Verwaltungsteile in einem sogenannten Testrahmen benötigt, müssen bei einer Messung die Zeitdauern, die diese Anweisungen benötigen, eliminiert werden.

Die Norm geht dazu in zwei Schritten vor. Im ersten Schritt werden die Programme in der Form gestartet, daß alle Anweisungen der Programme durchlaufen werden mit Ausnahme der zu testende Anweisung. Im zweiten Schritt werden die Programme, jetzt mit der zu testenden Anweisung oder Anweisungsfolge gestartet. Die für die zu testende Anweisung effektiv benötigte Ausführungszeit ergibt sich dann aus den Differenzen der Laufzeiten der Vordergrundprogramme und der Verweilzeiten.
Des weiteren ist in der Norm festgelegt, daß der vorgeschlagene Testrahmen zwar nicht zwingend in allen Einzelheiten übernommen zu werden braucht, jedoch muß er in allen Testreihen in der einmal gewählten Form beibehalten werden.

Zu den in der Norm beschriebenen Testfunktionen gibt es in einem Anhang eine beispielhafte Realisierung der Testprogramme in den Programmiersprachen PEARL und FORTRAN. Durch die Verwendung genormter Hochsprachen sind die Testprogramme mit geringen Änderungen auf unterschiedlichen Zielsystemen ablauffähig und erleichtern einen Vergleich.

3.2 Kurzbeschreibung der Testfunktionen

Das Hintergrundprogramm ist bei allen Testfunktionen das gleiche und besteht aus einem arithmetischen Ausdruck (eine Reihe von Ganzzahldivisionen) der in einer Schleife sehr oft wiederholt wird. Das Programm belegt den Zentralprozessor, wenn die Vordergrundprogramme ihn nicht benötigen.

Im Vordergrund laufen die verschiedenen Testfunktionen:

1. Multiplikation

   Es werden Fließkomma-Zahlen aus einem Feld mit Fließkomma-Zahlen aus einem zweiten Feld multipliziert und einem dritten Feld zugewiesen. Das Ergebnis ist dabei von derselben Genauigkeit wie die der Ausgangszahlen.

2. Bedingte Sprünge

   Hier wird der Zeitbedarf einer als wahr vorbelegten Bedingung und des nachfolgenden Sprunges ermittelt. Die Abfrage kann auf >, < oder = parametriert werden.

3. Suchen eines Bitmusters

   Nach einem vorgegebenen Algorithmus wird ein Feld von Bit(8)-Größen nach einem vorgegebenen Bitmuster an festgelegten Positionen innerhalb der Bit(8)-Werte abgesucht. Die eintreffende Erfolgsquote ist dabei parametrierbar.

4. Interrupt mit Programmstart

   Das zyklische einplanen einer höherpriorisierten Hilfstask durch das Hintergrundprogramm. Durch einen Zähler in der Hilfstask wird die Anzahl der Aktivierungen registriert. Messergebnis ist die Zentralprozessorzeit für Taskwechsel und erhöhen des Zählers.

5. Digital E/A

   Ausgabe und Einlesen eines 16-Bit-Wortes über die Digital E/A, wobei die Digital-Ausgabe direkt auf die Digital-Eingabe geschaltet wird. Die Kontrolle der einwandfreien Funktion kann durch Vergleich der Werte erfolgen.

6. Dialog am Terminal

   Ein festgelegter Satz von 50 Zeichen wird an ein Protokollgerät ausgegeben.

7. Matrixinversion

Es wird eine reguläre (n,n)-Matrix invertiert. Gemessen wird die Ausführungszeit. Jedoch ist die Rechengenauigkeit von größerer Bedeutung.

8. E/A-Operationen auf Peripherie-Speicher

Es werden Datensätze mit mindestens 4 Byte auf einen Peripheriespeicher geschrieben und gelesen. Die Ausführungszeiten der Treiber-Programme werden gemessen.

9. Synchronisation mit Botschaftenverkehr zwischen Tasks

Zwei Tasks senden und empfangen abwechselnd eine Botschaft und sind so zu synchronisieren, daß die eine Task jeweils wartet, bis die andere Task sendet, bevor sie selber sendet.

10. Datenübertragung mit störungsfreier Übertagungsstrecke

Zwischen zwei Datenübertragungseinheiten werden mit Hilfe von Übertragungssoftware Datensätze vorgebbarer Länge ausgetauscht.

11. Erzeugung von ablauffähigem Code plus Programmstart

Diese Testfunktion untersucht den Zeitbedarf für einen Übersetzungs-Binde- und Ladevorgang mit anschließendem Start des Programms. Die gemessene Zeit gibt Auskunft über Schnelligkeit der Programmerzeugung und ist ein nicht unerheblicher Aspekt bei der Programmentwicklung.

4. Testergebnisse mit RTOS-UH-PEARL auf FORCE Profikit 2

Zur Veranschaulichung seien zum Abschluß einige Meßergebnisse vorgestellt. Basis dieser Testreihe ist der Einplatinenrechner FORCE Profikit2 mit der CPU 68000-8. Das System besitzt keinen Hintergrundspeicher. Das Betriebssystem RTOS-UH Version FD und der PEARL-Compiler wurden an der Universität Hannover entwickelt und sind in EPROMs resident auf der Karte./3/

Mit den Testprogrammen wurden folgende Meßergebnisse erzielt. Die angegebenen Zeiten sind die jeweiligen Zentralprozessorzeiten.

Testfunktion:	Ausführungszeit	
Multiplikation	180	µsek
Bedingter Sprung	92	µsek

Suchen eines Bitmusters		
1% Erfolg	615	msek
10% Erfolg	623	msek
50% Erfolg	652	msek
99% Erfolg	689	msek
Interrupt mit Programmstart	734	µsek
Synchronisation	230	µsek
Digital E/A mit 16 Bit		
Ausgabe	336	µsek
Aus- und Eingabe	783	µsek
Eingabe	447	µsek
Dialog am Terminal		
Prozessorzeit	11	msek
Verweilzeit	586	msek
Erzeugen von Ablauffähigem Code		
Übersetzen	15,6	sek
Übersetzen,Laden und Starten(ohne Protokoll)	16,2	sek

5. Schlußbetrachtung

Das in der Norm beschriebene Verfahren hat sich in verschiedenen Tests bewährt. Die vorgeschlagenen Testfunktionen können beliebig ergänzt werden, so daß sich der Test unterschiedlichen Anwendungsprofilen anpassen läßt. Der Test ist besonders einfach bei Systemen mit höheren Programmiersprachen anzuwenden. Vorraussetzung ist allerdings eine Echtzeituhr, die vom Programm angesprochen werden kann.

Die Programme sind als Anhang zur Norm auf Datenträger über das DIN zum Selbstkostenpreis erhältlich.

Literatur:

/1/ Nehmer,J.:
Systemarchitektur von Realzeitsystemen; Informatik Spektrum Band 7 Heft 2,April 84;Springer-Verlag, Berlin

/2/ DIN 19 242 - Leistungstest von Prozeßrechensystemen,Zeitmessungen Teil 1 -14 ,Beuth-Verlag 1983

/3/ Gerth,W.:
Basic-PEARL für Mini- und Mikrorechner; PEARL-Rundschau Nr1,1981. PEARL-Verein,Stuttgart.

/4/ DIN 66 253 - Teil 1,Basic PEARL ,Beuth-Verlag 1981