

Autosar-Betriebssystem mit direkter Task-Laufzeitmessung

Gute Echtzeit-Embedded-Programmierung erfordert vom Software-Ingenieur viel Mühe und Aufwand, um die Klippen knapper Prozessor-Ressourcen zu umschiffen. Willkommene Hilfe bietet jetzt ein Embedded-Betriebssystem, das nicht nur Laufzeiten von Tasks und Interrupts direkt ermittelt, sondern in Verbindung mit dem Analysewerkzeug „TimingAnalyzer“ auch **LAUFZEITKONFLIKTE** aufdeckt.

Software Design ist einfach, wenn der Prozessor weit mehr Rechenzeit zur Verfügung stellt als die Anwendung benötigt. In der Praxis liegt jedoch üblicherweise der umgekehrte Fall vor. Moderne Embedded Software ist häufig komplex und besteht dabei aus mehreren, zeitlich unabhängig ablaufenden Prozessen (Tasks) und Interrupt-Service-Routinen (ISRs). Der Einsatz eines preemptiven Betriebssystems wie MICROSAR OS von Vector trägt wesentlich zu einer klaren Struktur, verkürzten Entwicklungszeiten und besserer Wartbarkeit bei. Ein stabiles System erfordert aber zusätzlich die Berücksichtigung der Wechselwirkung aller implementierten Tasks. Im Fokus stehen hier vor allem die Datenkonsistenz beim Zugriff auf gemeinsam genutzte Datenbereiche und das Laufzeitverhalten der einzelnen Tasks.

Laufzeitverhalten

Datenkonsistenz lässt sich mit Hilfe der vom Betriebssystem bereitgestellten Me-

chanismen sicherstellen. Für das korrekte Laufzeitverhalten ist aber der Entwickler verantwortlich. Er muss sicherstellen, dass alle Prozesse innerhalb ihres Zeitlimits abgearbeitet werden können.

Soll zum Beispiel ein Nutzer nach Betätigen eines Bedienelements innerhalb von 10 ms eine Wirkung erkennen, ist das ein typisches Beispiel für das Zeitlimit einer Task-Laufzeit. Meistens sind Entwickler jedoch mit zyklischen Vorgängen konfrontiert, bei denen jeder Vorgang zwingend vor Beginn des nächsten Zyklus abgeschlossen sein muss. Ebenso muss beim Software-Design berücksichtigt werden, dass eine laufende Task jederzeit durch einen Interrupt oder eine Task höherer Priorität unterbrochen werden kann.

Bild 1 verdeutlicht die Aufgabe anhand einer aus Task A und den Interrupts ISR 1 und ISR 2 bestehenden Anwendung. Die Task wird zyklisch alle 10 ms aufgerufen, benötigt eine Laufzeit von 5 ms und lastet den Prozessor damit zu 50% aus. Die Laufzeit der beiden Inter-

rupts beträgt jeweils 3 ms. Während die Unterbrechung der Task durch nur einen Interrupt unkritisch ist (Zyklus 2), führen beide Interrupts im gleichen Zyklus zu einer unzulässig langen Rechenzeit von 11 ms und damit zur Überschreitung des nächsten Startzeitpunkts (Zyklus 5). Solche Zeitkonflikte können zu sporadisch auftretenden und schwierig identifizierbaren Fehlern wie Datenverlusten, Timeouts, wahrnehmbaren Verzögerungen etc. führen. Eine gründliche Analyse ist daher vor Auslieferung der Software unverzichtbar.

Laufzeitmessungen direkt im Betriebssystem

Die Basis jeder Analyse sind genaue Kenntnisse über die Task- und Interrupt-Laufzeiten, die sich über verschiedene Verfahren ermitteln lassen. Eine einfache Lösung besteht darin, an Beginn und Ende der Tasks spezielle Testroutinen zum Toggeln von I/O-Ports einzufügen, die man zum Beispiel via externem Oszilloskop oder Logikanalysator beobachtet. Eine

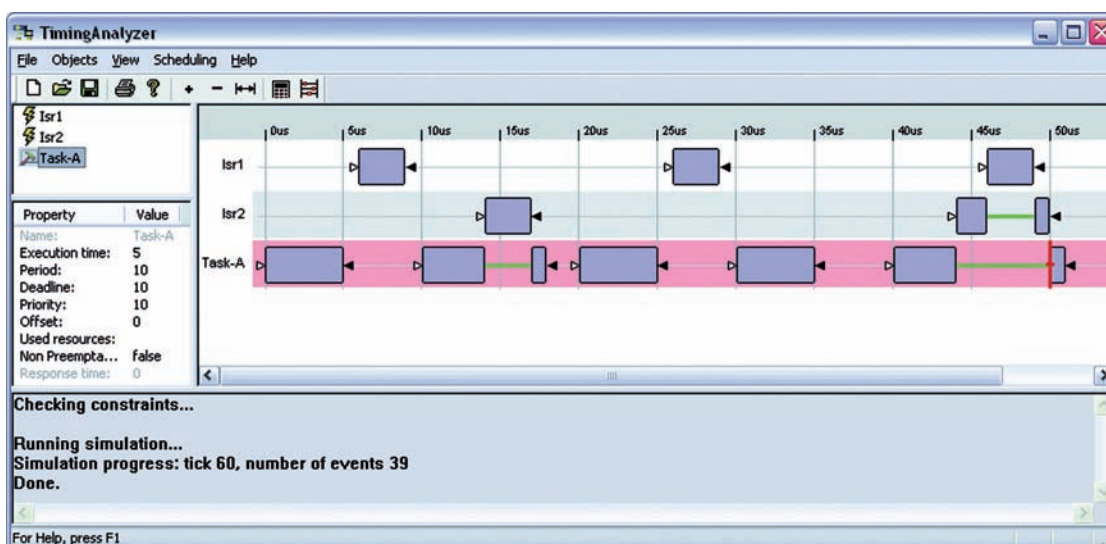


Bild 1: Das gleichzeitige Auftreten beider Interrupts führt zu einer unzulässigen Verzögerung von Task A.

andere Möglichkeit ist das Triggern spezieller Messroutinen, die über einen eigenen Timer jeweils Zeitstempel erzeugen und ihre Daten über eine (serielle) Datenschnittstelle an einen PC weitergeben.

Die Nachteile dieser Verfahren sind offensichtlich, denn einerseits ist der Code unter erheblichem Aufwand und Fehlergefahr wiederholt an mehreren Stellen manuell zu ändern, und andererseits umfasst das gemessene Zeitintervall neben der Laufzeit der untersuchten Routine auch die Laufzeiten aller unterbrechenden Tasks und ISRs.

MICRSOSAR OS

Ab sofort bietet MICRSAR OS eine Funktion zum Messen der Ausführungs- und Sperrzeiten ausgewählter Tasks und Interrupts direkt im Betriebssystem, die auch Unterbrechungen bei der Laufzeit-ermittlung berücksichtigt. Da nun keine Modifikationen des Anwendungscodes mehr erforderlich sind, vereinfacht sich die Arbeit des Testingenieurs signifikant. Man benötigt lediglich eine Testroutine zum Auslesen der Messdaten. Prinzipiell bietet sich dafür jede unterstützte serielle oder parallele Schnittstelle der Prozessor-beziehungswise Steuergeräte-Hardware an; ebenso aber auch ein Speicherabbild im Emulator. Über die Konfiguration des Betriebssystems schaltet der Anwender die Messfunktion bequem zu oder ab. Bild 2 verdeutlicht die Vereinfachungen.

Laufzeitanalyse

Nach den Laufzeitmessungen im Betriebssystem analysiert der Software-Ingenieur im nächsten Schritt die Messdaten und prüft, ob jede Task ihre Aufgabe innerhalb des geplanten Zeitrahmens abarbeiten kann. Für diesen Zweck liefert Vector mit dem Betriebssystem auch gleich das Analysewerkzeug „Timing-Analyzer“ aus.

Der TimingAnalyzer nutzt das Prinzip des Deadline Monotonic Scheduling. Der Ansatz basiert auf der Annahme, dass für jede Task ein Zeitlimit zur Abarbeitung ihrer Aufgabe existiert. Bei periodischen Tasks muss dieses Zeitlimit kleiner als die Periode sein. Aperiodische Tasks bildet das System durch periodische Tasks nach, bei denen die Periode der minimalen Inter-Arrival-Zeit entspricht. Unter Inter-Arrival-Zeit versteht man die Zeit zwischen zwei Task-Aktivierungen, zum Beispiel zwei Betätigungen des Blinkerhebels.

Der Software-Ingenieur füttert den TimingAnalyzer mit den gemessenen Laufzeiten der Tasks und ISRs und vervollständigt diese Daten durch Angabe von

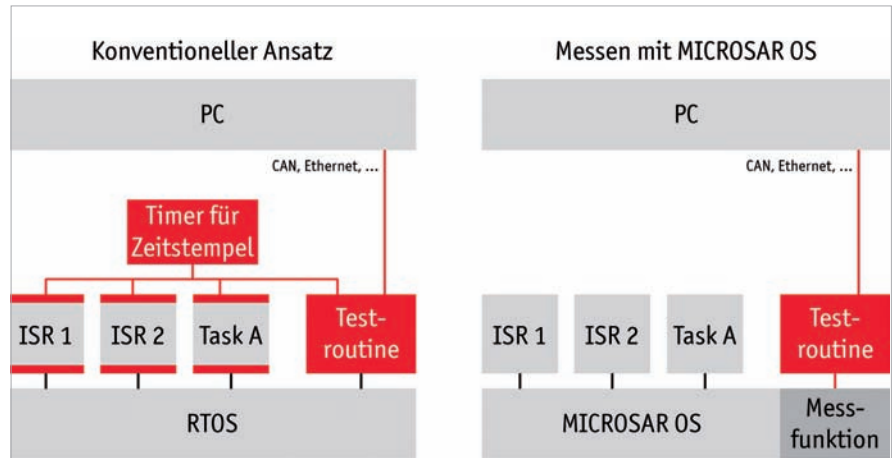


Bild 2: Methoden der Laufzeitmessung: Die rot gekennzeichneten Bereiche sind vom Anwender zu erstellende Software-Modifikationen.

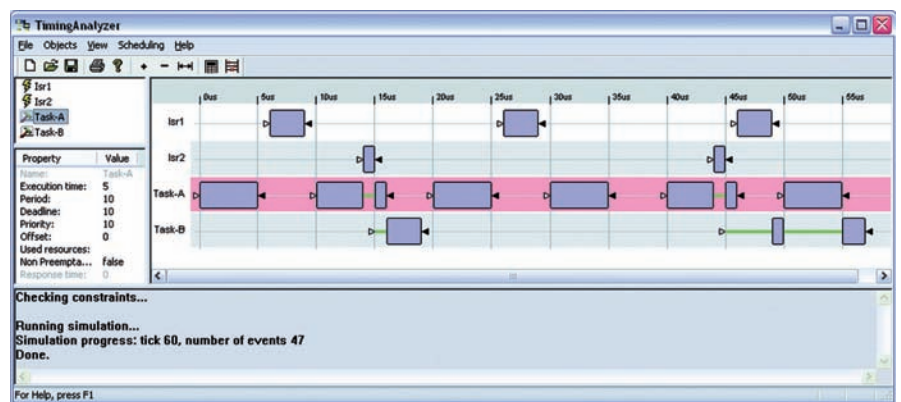


Bild 3: Lösung der Rechenzeitkonflikte durch Funktionsauslagerung.

Task-Prioritäten, einzuhaltenden Deadlines und typischen Zykluszeiten. Anhand dieser Vorgaben prüft der TimingAnalyzer nun, ob alle Tasks ihre Zeitlimits jederzeit einhalten und stellt das berechnete Zeitverhalten grafisch dar (siehe Abbildungen 1 und 3). Der Analyse-Algorithmus berücksichtigt dabei auch die speziellen Abhängigkeiten von Tasks und Interrupts im Rahmen der gemeinsamen Nutzung von Hardware- oder Software-Ressourcen.

Bild 3 zeigt eine mit Hilfe des Timing-Analyzers gefundene Lösung für das Beispiel aus Abbildung 1. Man teilt ISR 2 auf in eine Kernfunktion mit einer Laufzeit von 1 ms auf Interruptlevel und in eine niederpriorie Nachbearbeitung mit einer Laufzeit von 4 ms auf Tasklevel. Eine solches Szenario ist mit geringem Zeitbedarf im TimingAnalyzer eingerichtet und das Resultat innerhalb weniger Sekunden verfügbar. Der Entwickler prüft auf diese Art und Weise schnell und einfach die Auswirkungen größerer Änderungen des Source-Codes oder neuer Konfigurationen.

Ausblick

Die erweiterten Möglichkeiten von MICRSAR OS bilden zusammen mit dem TimingAnalyzer ein effizientes Duo zur Embedded Software Entwicklung. Während das Betriebssystem die Bestimmung der Ausführungszeiten von Tasks und Interrupt Service Routinen vereinfacht, dient das Analyse-Werkzeug zur grafischen Darstellung und zum Prüfen auf Laufzeitkonflikte. Zurzeit werden die Messdaten über eigene Messroutinen erfasst und manuell in das Analysewerkzeug übertragen. Für die Zukunft ist es geplant, die Daten mit Hilfe standardisierter Protokolle wie XCP automatisch auf den PC und in den TimingAnalyzer zu laden.

Dr.-Ing. Helmut Brock ist Produktmanager für OSEK/VDX- und Autosar-Betriebssysteme bei Vector Informatik

infoDIRECT www.all-electronics.de
 Link zu Vector Informatik 321AEL0110

Alle Bilder: Vector Informatik GmbH