

Softwareentwicklung mit mehreren Teams

Paralleldesign

In der Automobilindustrie ist das Multitasking-Betriebssystem OSEK/VDX weit verbreitet. Obwohl es als kleines und schlankes System konzipiert ist, wird es durchaus für komplexe Aufgaben eingesetzt. Dabei arbeiten häufig verschiedene Teams an der Softwareerstellung eines Projekts. Die Koordination der Betriebssystemkonfiguration erweist sich dabei als Problem. Deshalb wird hier ein Verfahren vorgestellt, welches die parallele Implementierung von Teillösungen erlaubt und die Integration zum Gesamtprojekt erleichtert.

OSEK/VDX ist ein präemptives Multitasking-Echtzeitbetriebssystem. Die Spezifikation wurde aus den Anforderungen an Steuergeräte im Kraftfahrzeug hergeleitet. Sie lässt sich aber auch auf viele andere Anwendungsbereiche übertragen, bei denen ähnliche Anforderungen gelten:

- Schnelle Reaktionszeiten, d.h. minimale Rechnerbelastung durch das Betriebssystem
- Verwendung nur des internen RAM
- Sehr kleiner ROM-Bedarf

Diese Anforderungen erfüllt ein Konzept mit den folgenden Eigenschaften:

- Beschränkung auf die notwendigen Grundfunktionen
- Skalierbarkeit, d.h. nicht verwendete Funktionen können weggelassen werden

- Statisches Verhalten
- Offener Betriebssystemstandard

Prioritäten, Prozesse und Interrupts

Das wichtigste Element eines OSEK/VDX-Betriebssystems ist der »Task«, womit die einzelnen Anwendungsprozesse bezeichnet werden. Diese Tasks haben definierte Eigenschaften wie z.B. eine Priorität oder die Unterbrechbarkeit (Preemptiveness), welche getrennt für jede einzelne Task eingestellt werden kann. Selbstverständlich ist auch die Einrichtung von Interruptservice-Routinen (ISRs) möglich, wobei man wählen kann, ob die Verwendung von Betriebssystemdiensten innerhalb der ISR möglich sein soll.

Die einzelnen Teilprozesse lassen sich durch die Elemente »Event« oder »Resource« synchronisieren. Ein Event ist ein binäres Signal, mit dem sehr elegant der Ablauf einer Task gesteuert werden kann. In OSEK/VDX

kann eine Task mehrere verschiedene Events empfangen und unterscheiden.

Die »Ressource« regelt den Zugriff mehrerer Task auf eine gemeinsam genutzte Komponente wie z.B. ein Datenarray. Sie kann mit einer Semaphore verglichen werden, geht aber im Funktionsumfang darüber hinaus.

So sind Deadlocks und Prioritätsumkehr bei der OSEK/VDX-»Ressource« ausgeschlossen. Anstelle des Datenaustausches über eine globale Speicherzelle bietet OSEK/VDX einen Datenübertragungs-Mechanismus im Betriebssystem, genannt »Message«. Hiermit lassen sich Daten in gepufferter oder einfacher Form an mehrere Empfänger weiterleiten.

Eingebaute Alarmanlage

Ein weiteres wichtiges Element sind die Alarme. Bei Erreichen eines Sollwertes in einem Referenzzähler wird eine Task aktiviert. Dies kann durch einen expliziten Start oder das Versenden eines Events geschehen. Der wichtigste Anwendungsfall für Alarme ist das Erzeugen von einzelnen oder zyklischen Zeitintervallen.

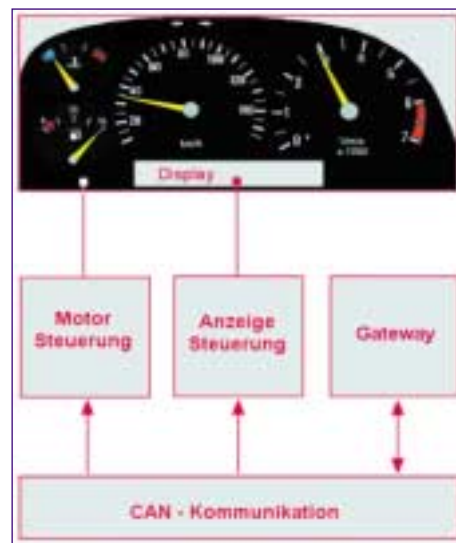


Bild 1 Eine Anwendung – mehrere Softwaremodule

Die einzelnen Systemkomponenten besitzen Eigenschaften, welche sich an die Anforderungen der jeweiligen Anwendung anpassen lassen. Diese liegen zum Teil statisch fest, andere ändern sich während der Laufzeit. Als Beispiel sei eine blinkende LED angeführt. Die Betriebssystem-Elemente benötigen eine Task zum Setzen des Portpins und einen Alarm als Zeitgeber. Da die Blinkfrequenz variieren kann, wird das Zeitintervall im Anwendungsprogramm über einen Betriebssystemdienst dynamisch gesetzt. Die Task, welche nach Ablauf des Intervalls starten soll, weist man hingegen dem Alarm noch vor dem Compilieren als feste Eigenschaft zu.

Diese statische Zuordnung vereinfacht die Verwaltungsabläufe im Betriebssystem erheblich.

Die Hersteller von OSEK/VDX-Betriebssystemen bieten grafische Oberflächen, mit denen der Entwickler derartige Konfigurationen komfortabel und eigenständig vornehmen kann. Eine Eigenschaft

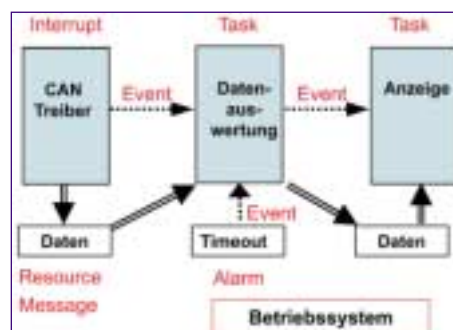


Bild 2: Die Elemente von OSEK/VDX

Dr. Helmut Brock ist bei Vector Informatik für das Produktmarketing OSEK/VDX zuständig

von OSEK/VDX ist es, dass die Konfiguration für das gesamte Projekt in einem standardisierten Format abgelegt wird.

Diese Konfigurationsdatei ist die Grundlage für die Generierung der Betriebssystem-Elemente. Eine solche Datei enthält die eingerichteten Systemelemente zusammen mit ihren Attributen. Diese Datenbeschreibung geschieht in einem standardisierten Format, der »OSEK Implementation Language«, kurz OIL genannt. Es handelt sich dabei um einen im Klartext lesbaren ASCII-Code.

Die standardisierte Ablage gewährleistet eine Portierbarkeit im Falle eines Prozessorwechsels. Lediglich die hardware-spezifischen Definitionen sind in solch einem Fall anzupassen.

Betriebssystem im Teameinsatz

Wie bereits erwähnt, ist für jedes Projekt eine Konfigurationsdatei notwendig. Diese wird sinnvollerweise zentral verwaltet und gepflegt. Solch ein Verfahren entspricht aber nicht der Arbeitsweise bei der Steuergeräte-Entwicklung. Hier arbeiten häufig größere Entwicklergruppen, welche Module in getrennten Teams erstellen. Dieser Arbeitsweise sollte die Verwaltung der Konfigurationsdateien entsprechen, um ein unabhängiges Arbeiten zu ermöglichen.

Zum einen sind geringfügige Anpassungen bei einer zentralen Verwaltung umständlich, außerdem gibt es bei Tests von Teilsystemen Schwierigkeiten, da die Kon-

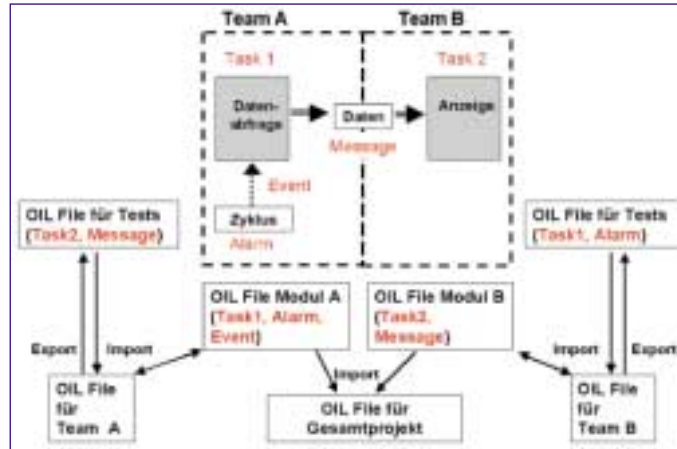


Bild 3 Beispiel für das Arbeiten mit dem Komponentenmanagement

figurationsdatei Systemelemente definiert, welche in diesen Modulen nicht enthalten sind. Fehlermeldungen des Compilers sind dann das Resultat.

Ein anderer Ansatz ist der Entwicklungsablauf mit dezentraler Konfiguration. Hier erstellt jede Arbeitsgruppe eine eigene Konfigurationsdatei, welche nur die Betriebssystemelemente des jeweiligen Moduls berücksichtigt. Im Rahmen der Softwareintegration werden diese Teilkonfigurationen zur OIL-Datei des Gesamtprojekts zusammengefasst. Dies entspricht eher dem Arbeitsfluss und erlaubt das Zusammenstellen von Anwendungen aus vorhandenen Softwaremodulen.

Arbeiten mit verschiedenen Modulen

Um die Übersicht zu behalten und auch um die Teilkonfigurationen gezielt erstellen und einbinden zu können, wird ein neues Attribut für die Betriebssystemelemente eingeführt: das Attribut »Component«.

Dabei ist zu bedenken, dass es sich um eine gemäß der OSEK/VDX-Spezifikation zulässige aber nicht standardmäßige Erweiterung handelt. Das Attribut »Component« lässt sich über das mit dem Betriebssystem mitgelieferte Konfigurationstool hinzufügen und auch wieder entfernen. Somit ist die Portabilität zu anderen OSEK/VDX-Betriebssystemen jederzeit gegeben. Richtet man neue Betriebssystemelemente ein, z.B. eine weitere Task, so setzt man für diese zusätzlich das Attribut »Component«. Der Anwender wählt den Attributwert, d.h. den Modulnamen, aus einer vorher von ihm erstellten Liste aus. Damit ist eine einheitliche Namensgebung gewährleistet. Die resultierende OIL-Datei wird abschließend unter dem Modulnamen abgespeichert.

Quer verknüpft

Es kann durchaus vorkommen, dass in einem Modul Systemelemente definiert sind, welche mit Elementen

in anderen Modulen verknüpft sind.

In Bild 3 wird z.B. in Modul A eine Message definiert, die einer Task geschickt werden soll, welche in Modul B konfiguriert ist. Dies lässt sich durchaus in einem Arbeitsgang erstellen.

Man muss lediglich für die Message das Attribut »Modul A« und für die Task das Attribut »Modul B« wählen. Über eine Exportfunktion lassen sich die beiden Elemente getrennt in modul-spezifischen Konfigurationsdateien ablegen.

Die doppelte Konfiguration der Elemente, wie sie in Bild 3 dargestellt ist, ist nicht notwendig.

Es ist lediglich wichtig zu wissen, dass das unabhängige Entwickeln einzelner Module möglich ist.

Softwareentwicklung in der Praxis

Mit dem Komponentenmanagement von Vector Informatik lässt sich ein OSEK/VDX-konformes Betriebssystem auf einfache Weise bei der Softwareentwicklung in getrennten Teams nutzen.

Die Implementierung von Modulen und ihre Integration zu Gesamtanwendungen wird durch das Komponentenmanagement einfacher und entspricht besser dem realen Entwicklungsablauf, der ja nur in den seltensten Fällen monolithisch geschieht. Modulare Programme werden eben auch modular entwickelt. (mc)

Vector Informatik

Telefon 07 11/80 67 00

Fax 07 11/80 67 01 11