

ProOSEK: Vom OSEK/VDX-Standard zum Echtzeit-Betriebssystem

Jochen Schoof

3SOFT GmbH, Wetterkreuz 19a, 91058 Erlangen
Jochen.Schoof@3SOFT.de

Der OSEK/VDX-Standard¹ definiert ein Betriebssystem für den Einsatz in embedded real-time Geräten. Im wesentlichen wird dabei aber nur eine Programmierschnittstelle (API) beschrieben. Zu den für den Anwender wichtigen Fragen des Zeit- und Ressourcenverbrauchs äußert sich der Standard nicht. Folglich spielt für die Konformität einer Implementierung weder ihr zeitliches Verhalten noch ihr Umgang mit den Ressourcen eine Rolle.

Im Automobilssektor als dem Haupteinsatzgebiet von OSEK/VDX-konformen Systemen werden überwiegend 16-Bit Microcontroller in Single-Chip Konfiguration benutzt. Typische Speichergrößen bewegen sich zwischen 32 und 128 KB ROM bzw. zwischen ein und zehn KB RAM. Die Prozessoren werden mit Frequenzen zwischen vier und acht Megahertz getaktet, wobei man oft gezwungen ist, niedrigere Werte einzuhalten, um die Abstrahlung möglichst gering zu halten. Hinzu kommt, dass neben dem Betriebssystem noch weitere standardisierte Module zu integrieren sind (z.B. Kommunikation, Diagnose). Die Optimierung des Betriebssystems im Hinblick auf Zeit- und Ressourcenverbrauch ist also von zentraler Bedeutung.

Da OSEK/VDX eine individuelle Konfiguration des Betriebssystems für jede Applikation vorsieht, kann bei dieser Optimierung auch Wissen über deren genaue Anforderungen genutzt werden. Entscheidend hierfür ist ein Konfigurationswerkzeug, das in der Lage ist, die jeweiligen Optimierungspotenziale zu erkennen und auszunutzen. Nur so können ohne die direkte Beteiligung des Anwenders Verbesserungen erreicht werden.

Ein gutes Beispiel für das vorhandene Optimierungspotenzial ist der Scheduler: Der Bogen spannt sich von Applikationen mit einer einzigen single-shot Task bis zu solchen mit einer Vielzahl eventgesteuerter, präemptiver und mehrfach aktivierbarer Tasks. Zwischen diesen beiden Extremen mit sehr unterschiedlichen Anforderungen gibt es eine ganze Reihe sinnvoller Abstufungen. So definiert OSEK/VDX bereits vier so genannte Conformance Classes und drei grundlegende Scheduling-Strategien, die eine Skalierung in zwölf Schritten erlauben. In der Praxis zeigt sich aber, dass sogar eine noch feinere Anpassung möglich ist. So ist zum Beispiel das vom Standard hergestellte Junktim zwischen mehrfach aktivierbaren Tasks und mehreren Task gleicher Priorität in vielen Fällen nicht besonders sinnvoll.

Am Beispiel von ProOSEK, einer in der Automobilindustrie bereits im Einsatz befindlichen OSEK-Implementierung, werden mögliche Strategien zur Optimierung des OSEK-Kernels vorgestellt. Als Ergebnis des Einsatzes dieser Techniken sind minimale Kernelgrößen von weniger als einem KB erreichbar.

1. Kostenlos zu beziehen unter der URL http://www.osek-vdx.org/osekvdx_os.html.