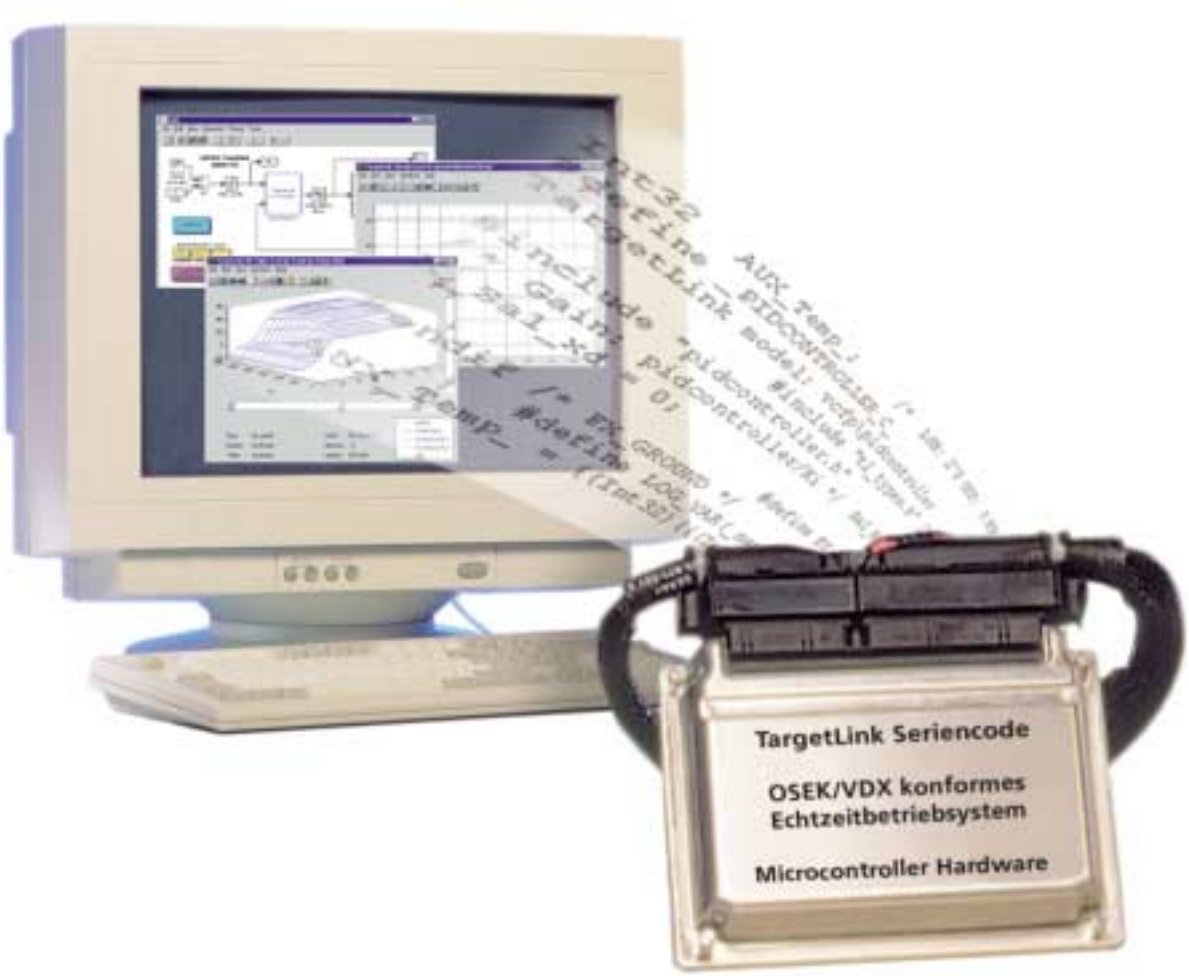
Von der Funktion zum serienreifen Steuergerät

Wie können neue Standards die Weiterentwicklung von Entwicklungswerkzeugen fördern und die Zusammenführung von bisher völlig getrennten Werkzeugen ermöglichen? Welchen Einfluss nehmen Echtzeitbetriebssystem-Standards wie OSEK/VDX auf die hardwarenahe Codegenerierung? Diese und viele andere Fragen stellen sich heute die Entwickler von Steuergeräten, die sich mit neuen Methoden der Software-Entwicklung auf diesem Gebiet beschäftigen. Wie die dSPACE GmbH die vollständige Integration eines Echtzeitbetriebssystems in den Codegenerierungsprozess verfolgt wird hier anhand eines Gesamtüberblicks beschrieben.



1 Einleitung

Die Software-Entwicklung für Steuergeräte wird in zunehmendem Maße mit modellbasierenden Entwicklungstools wie MATLAB/Simulink/Stateflow von The MathWorks durchgeführt. Steht die Spezifikation unter Zuhilfenahme dieser Tools fest, beginnt die Arbeit der Codierung. Die grafisch beschriebenen Funktionen müssen nun in C-Code umgewandelt und zusammen mit einem Echtzeitbetriebssystem auf der Steuer-gerätehardware implementiert werden.

2 TargetLink für die Codegenerierung nach Maß

Ein Seriencodegenerator wie beispielsweise TargetLink von dSPACE löst das anstehende Codierproblem. TargetLink ergänzt die Simulink-Umgebung um eine Codegenerierung in Serienqualität und ist in der Lage, portablen ANSI-C-Code sowie auch prozessor- und compilerspezifischen Code zu generieren.

TargetLink kann sowohl ein unverändertes Simulink-Modell vollautomatisch – also mit wohlüberlegten Standardeinstellungen – in eine optimale Echtzeitimplementierung umsetzen, als auch umfangreiche zusätzliche Optionen anbieten, die eine exakte Anpassung an spezielle Anforderungen erlauben. Dabei ist zu beachten: Ein Simulink-Modell kennt ursprünglich keine Tasks, sondern ist aus anderen Objekten aufgebaut, die der Codegenerator auf Betriebssystemeinheiten abbilden muss. Für die automatische Codegenerierung aus Simulink für Echtzeitbetriebssysteme ergeben sich daher zwei wesentliche Ziele:

- Die Abbildung von etablierten Simulink-Modellierungskonzepten auf verfügbare Betriebssystemobjekte.
- Die Schaffung neuer Spezifikationsmöglichkeiten auf Modellebene für wichtige Betriebssystemeigenschaften.

Eine leistungsfähige Codegenerierung kann die technischen Hürden zwischen der Modellierung und der Implementierung auf Basis eines Echtzeitbetriebssystems signifikant abbauen. Doch welche Rolle spielt dabei der OSEK/VDX-Betriebssystemstandard in Bezug auf TargetLink? Unter OSEK/VDX sind offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug/Vehicle Distributed eXecutive.

dSPACE erweitert in der nächsten Version von TargetLink den Anwendungsbereich der Codegenerierung um die Unterstützung von OSEK/VDX-konformen Betriebssystemen. Die Gründe hierfür sind:

- Der OSEK/VDX-Standard definiert eine einheitliche Betriebssystemfunktionalität und eine einheitliche Programmierschnittstelle der Betriebssystemdienste. Damit ist es einem Codegenerator möglich, auch den Anschluss an diese Softwareschicht zu automatisieren. Dies ist bei proprietären Echtzeitbetriebssystemen nur über zusätzliche manuelle Programmierung möglich.
- OSEK/VDX hat eine weite Verbreitung innerhalb der Automobilindustrie gefunden und es existiert inzwischen eine Reihe hocheffizienter, kommerzieller Betriebssysteme, die diesem Standard genügen. Folglich möchten die Anwender auch eine automatische Anbindung für die Codegenerierung an ihr OSEK/VDX-konformes Echtzeitbetriebssystem.

3 OSEK/VDX: Konzepte und Ziele

Der OSEK/VDX-Betriebssystemstandard beschreibt ein statisches Echtzeitbetriebssystem, dessen Objekte (Tasks, Events, Messages, Ressourcen, etc.) alle zur Compile-Zeit angelegt werden. Diese Betriebssystem-Objekte werden in einer speziellen Beschreibungssprache (OIL: OSEK/VDX Implementation Language) definiert. Zu einem Objekt gehören bestimmte OIL-Attribute wie beispielsweise die Priorität einer Task. Zusätzlich zu den Standard-Attributen können Hersteller von OSEK/VDX-konformen Betriebssystemen weitere Attribute definieren und so zum Beispiel ihr Betriebssystem auf bestimmte Mikrocontroller anpassen.

3.1 Ziele von OSEK/VDX

Ziel des OSEK/VDX-Betriebssystemstandards ist es, die Wiederverwendbarkeit und Portierbarkeit von Anwendungs-Software zu verbessern. Um dieses Ziel zu erreichen, wurden von den OSEK/VDX-Arbeitsgruppen Schnittstellen in den Bereichen Echzeitbetriebssystem, Kommunikation und Netzwerk-Management spezifiziert. Es soll hier nur auf den Bereich Echtzeitbetriebssystem eingegangen werden.

OSEK/VDX selbst ist also kein Betriebssystem, sondern ein Standard, der die Schnittstellen und das Verhalten eines OSEK/VDX-konformen Betriebssystems, im Folgenden OSEK/VDX-Betriebssystem genannt, definiert. Die Vorteile des OSEK/VDX-Standards sind:

- Verkürzung der Entwicklungszeit
- Erhöhte Software-Qualität durch Wiederverwendung bereits getesteter Software-Module
- Optimale Speicherausnutzung durch Skalierbarkeit

Die Autoren

Dr. Ing. Lutz Köster ist Gruppenleiter der Software-Entwicklung bei der dSPACE GmbH, Paderborn.

Dr. Ing. Rainer Otterbach ist Leiter Produktmanagement bei der dSPACE GmbH, Paderborn.

Dipl.-Ing. (FH) Günther Gruhn ist Technischer Redakteur bei der dSPACE GmbH, Paderborn.

Summary

Function Design to Production ECU

Is it possible to map operating system functionalities while during function design for an electronic control unit (ECU)? How can new standards promote the further development of development tools and enable completely separate tools to be brought together? What effect are real-time operating system standards such as OSEK/VDX having on hardware-related code generation? These and many other questions today face the developers of ECUs as they investigate new methods of software development for their field. The following overview gives an account of how dSPACE GmbH is striving for complete integration of a real-time operating system in the code generation process.

4.1 Die Schritte vom Simulink-Modell zur ausführbaren Task

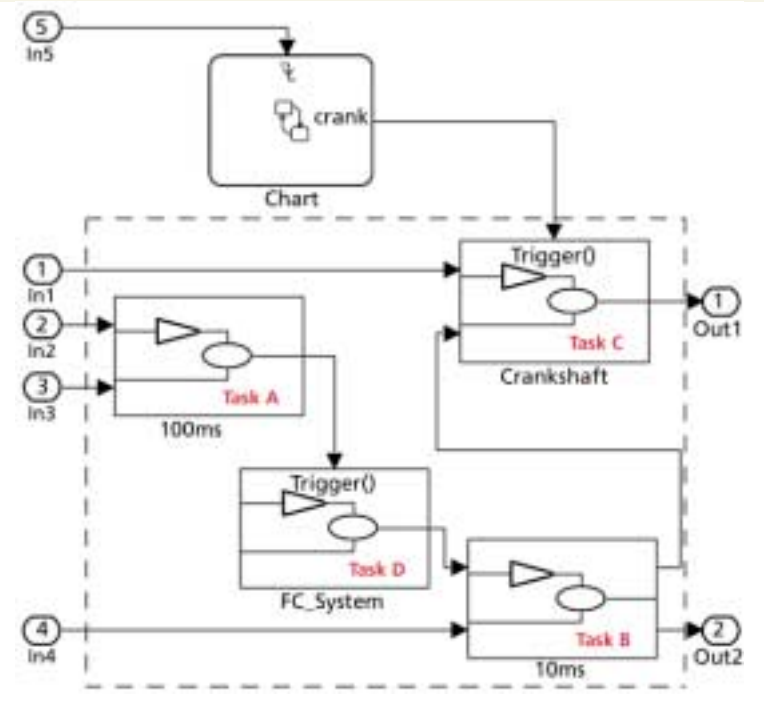


Bild 1: Typisches Simulink/Stateflow-Diagramm

- Wiederverwendung von Software in anderen Umgebungen.

3.2 Die Skalierbarkeit innerhalb von OSEK/VDX

Unterschiedliche Anwendungen erfordern unterschiedliche Betriebssystemfunktionalitäten. Um keine Steuergeräte-Ressourcen zu verschwenden, ist es notwendig, das Betriebssystem zu skalieren, also dessen Umfang an die Anwendung anzupassen.

Um dieses Ziel zu erreichen, sieht der OSEK/VDX-Standard zwei Stufen für eine schrittweise Fokussierung der Anwenderbedürfnisse vor:

- Mit einer Entscheidung für eine bestimmte der vier vordefinierten Conformanceklassen, die unterschiedliche Funktionsumfänge beschreiben.
- Mit individuellen Einträgen in die OIL-Datei. Diese statische Definition der Betriebssystemobjekte erlaubt die Generierung eines Betriebssystemkerns mit dem System-Generator des Betriebssystemherstellers, der genau nur die Funktionen enthält, die für die jeweilige Anwendung benötigt werden.

Der OSEK/VDX-Betriebssystemstandard definiert als wichtigste Objekte:

- Tasks, die als Software-Einheiten die einzelnen Funktionen der Steuergeräte-Software beinhalten.
- Counter, Alarme und Events, um wiederkehrende Ereignisse zu verarbeiten oder Tasks zu synchronisieren.
- Eine Message-basierende Inter-Task-Kommunikation für den Datenaustausch sowohl zwischen Tasks auf demselben Steuergerät als auch zwischen Tasks, die sich auf verschiedenen Steuergeräten befinden.
- Verriegelungsmechanismen (Ressourcen), um den gemeinsamen Zugriff von verschiedenen Tasks oder Interrupt Service Routinen (ISR) auf kritische Bereiche zu regeln.

4 Modellierung von Echtzeit-Anwendungen in Simulink/TargetLink

Eine Anwendung, die auf einem Echtzeitbetriebssystem basiert, ist in unterschiedliche Software-Einheiten aufgeteilt. Diese sogenannten Tasks fassen Software-Abschnitte mit gemeinsamen Echtzeitanforderungen wie Timing, Priorität etc. zusammen. Sie werden vom Echtzeitbetriebssystem periodisch zu definierten Zeiten aufgerufen oder durch aperiodische Hard- oder Software-Ereignisse aktiviert.

4.1 Die Schritte vom Simulink-Modell zur ausführbaren Task

Simulink nutzt zur hierarchischen Strukturierung des Modells sogenannte Subsysteme, die zu einer gewissen Funktionalität gehörende Basisblöcke zusammenfassen. Diese Basisblöcke werden entweder zyklisch mit einer bestimmten Abtastrate oder ereignisgetrieben abgearbeitet.

Bild 1 zeigt ein typisches Simulink/Stateflow Diagramm, das periodische und ereignisgetriebene Modellteile enthält. Um dieses Modell als OSEK/VDX-Echtzeitanwendung zu implementieren, bedarf es mehrerer Schritte.

4.2 Periodische Tasks

Im Beispiel von Bild 1 würde TargetLink das 10ms- und 100ms-Subsystem jeweils als separate Task (A und B) implementieren. Die zugehörigen C-Funktionen werden automatisch durch entsprechende Schlüsselwörter im Code als Task gekennzeichnet. Zusätzlich wird das Betriebssystem durch die Vorgaben von TargetLink so konfiguriert, dass diese Tasks automatisch in der korrekten Periodendauer aufgerufen werden. Hierfür werden OSEK-Alarme definiert, geeigneten Timern zugeordnet und in einer speziellen Startup-Routine durch entsprechende Betriebssystemaufrufe initialisiert.

4.3 Aperiodische Tasks

Ereignisgesteuerte Funktionen werden in Simulink durch getriggerte Subsysteme modelliert. Ihre Triggerung wird entweder durch bestimmte Bedingungen ausgelöst, die im Steuergerätecode selbst ausgewertet werden (FC_System in Bild 1), oder durch externe Ereignisse wie zum Beispiel Hardware-Interrupts.

Die intern getriggerten Systeme werden von TargetLink entweder als C-Funktionen, die nach Auswertung der Trigger-Bedingung direkt aufgerufen werden, implementiert oder optional – wie in Bild 1 – als separate Task D, die nach Auswertung der Trigger-Bedingung durch entsprechende Betriebssystemfunktionen aktiviert wird.

Extern getriggerte Systeme sind solche, deren Triggerung nicht durch Steuergerätecode ausgelöst wird, sondern zum Beispiel durch einen Hardware-Interrupt (Crankshaft in Bild 1). Für die Simulation können diese externen Triggerquellen mit Simulink-Blöcken nachgebildet (Chart in Bild 1) werden. Seriencode soll für diese Blöcke allerdings nicht generiert werden. Der Wirkungsbereich des Codegenerators

umfasst daher nur den im Bild 1 gestrichelten Bereich. TargetLink implementiert extern getriggerte Systeme standardmäßig als Tasks und stellt dafür Aktivierungsfunktionen zur Verfügung, die der Anwender den Triggerquellen zuordnen muss.

4.4 Vermeiden von Task-Overhead

Ein Steuergerät enthält üblicherweise eine größere Anzahl von Funktionalitäten (zum Beispiel Drosselklappenregelung, Leerlaufregelung, Abgasrückführung etc.), die häufig von verschiedenen Teams entwickelt und sinnvollerweise als getrennte Subsysteme dargestellt werden. Jedes dieser Features kann wiederum Teile enthalten, die zum Beispiel im 10ms-Raster oder synchron zur Kurbelwelle abgearbeitet werden müssen. TargetLink kann, sofern gewünscht, alle periodischen Subsysteme gleicher Abtastrate zu gemeinsamen Tasks zusammenfassen. Getriggerte Subsysteme mit gemeinsamer Triggerquelle werden ebenfalls zusammengefasst. Dabei spielt es keine Rolle, wie die Subsysteme im Modell verteilt sind. Die Zusammenfassung dieser verteilten Einzelsysteme gleicher Periodizität oder Eventzugehörigkeit – häufig Prozesse genannt – hilft Task-Overhead zu sparen.

Neben der automatischen Taskpartitionierung kann der Anwender diese auch gezielt selbst spezifizieren. Ein spezieller „Taskblock“, **Bild 2**, der in das Simulinkmodell eingefügt wird, erlaubt es dem Anwender, ein Simulink-Subsystem explizit einer schon existierenden oder einer neu anzulegenden Task zuzuordnen.

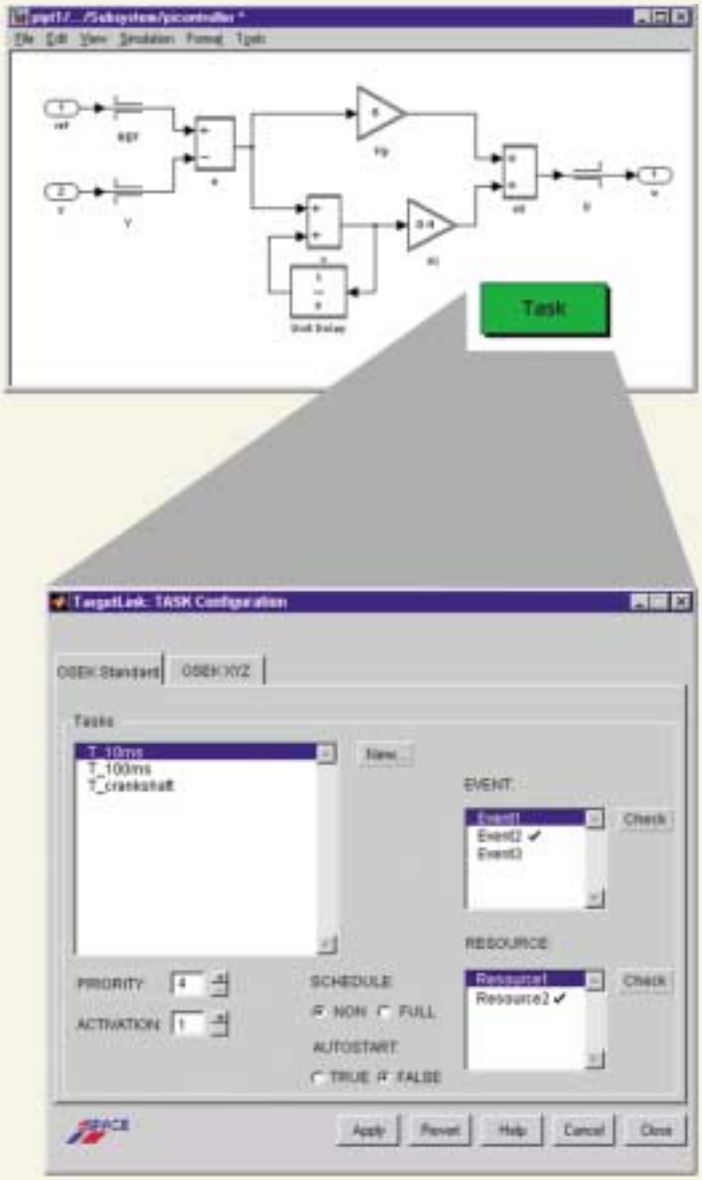


Bild 2: Taskblock-Dialog

4.5 OSEK/VDX konforme Taskkonfiguration

Mit der Aktivierung des Taskblocks (Doppelklick) gelangt man in den Taskblock-Dialog. Eine Liste zeigt hier die schon im Gesamtsystem vorhandenen Tasks an. Das aktuelle Subsystem kann einer dieser Tasks zugeordnet werden oder es kann eine neue Task kreiert werden. Dazu gibt es Menüeinträge für alle von OSEK vorgeschriebenen Attribute wie Priorität, Anzahl der Aktivierungen, Unterbrechbarkeit, benutzte Ressourcen oder zugehörige Events. Darüber hinaus können für eine Reihe von kommerziellen OSEK-Betriebssystemen auch die nicht standardisierten, herstellerspezifischen Optionen direkt aus der TargetLink-Umgebung eingestellt werden.

4.6 Inter-Task-Kommunikation und OIL-Datei

Der nächste wichtige Schritt ist die Umsetzung der Datenverbindungen zwischen den Tasks. Betriebssysteme bieten üblicherweise Dienste zum geschützten Datenaustausch zwischen Tasks an. OSEK/VDX definiert dafür sogenannte Messages, die allerdings für den Datenverkehr zwischen Tasks auf demselben Steuergerät nicht immer die effizienteste Lösung darstellen.
Daher kann TargetLink auch automatisch den jeweils effizientesten Weg des Datenaustauschs, der abhängig von den Taskeinstellungen ist, implementieren (globale Variablen mit lokalen Kopien oder ohne lokale Kopien, Interrupt-Sperre, Ressourcen-Belegung etc.).

Mit den Einstellungsdaten der Taskkonfiguration und den Einstellungsdaten der Inter-Task-Kommunikation wird von TargetLink eine Offline-Beschreibung der gesamten Anwendung in Form einer OIL-Datei generiert, **Bild 3**. Mit den hierin enthaltenen Informationen kann der System-Generator des Betriebssystemherstellers einen optimal angepassten Echtzeitkern erzeugen.

5 OSEK/VDX-Konfiguration und verschiedene Werkzeuge

Ein typisches Steuergeräteprogramm besteht aus mehreren C-Quelldateien, von

4.6 Inter-Task-Kommunikation und OIL-Datei

```
TASK T_10ms
{
  PRIORITY = 4;
  SCHEDULE = NON;
  ACTIVATION = 1;
  AUTOSTART = FALSE;
  RESOURCE = Resource2;
  EVENT = Event2;
};

TASK T_100ms
{
  PRIORITY = 3;
  SCHEDULE = FULL;
  ACTIVATION = 1;
  AUTOSTART = FALSE;
  RESOURCE = Resource2;
};

TASK T_crankshaft
{
  PRIORITY = 3;
  SCHEDULE = FULL;
  ACTIVATION = 1;
  .
  .
```

Bild 3: Beispiel einer OIL-Datei

5 OSEK/VDX-Konfiguration und verschiedene Werkzeuge

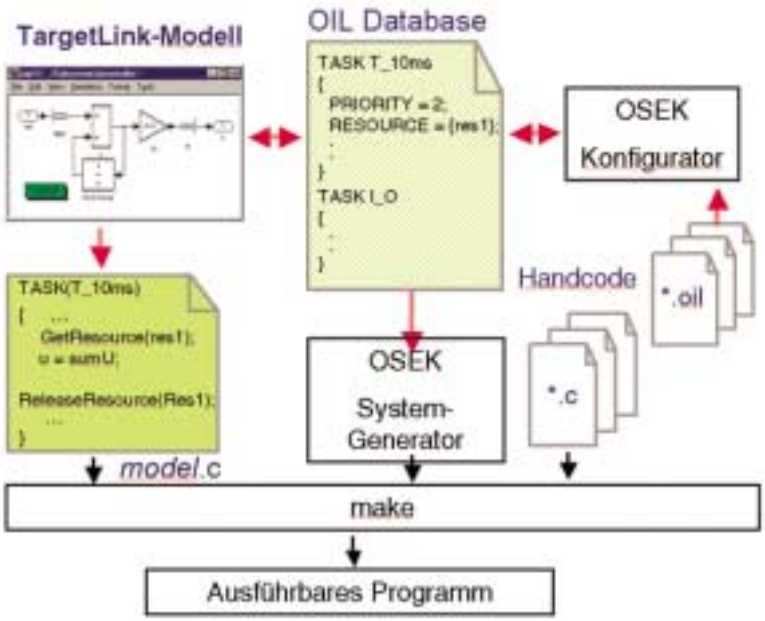


Bild 4: Zusammenspiel der Entwicklungswerkzeuge

denen einige beispielsweise von TargetLink erzeugt, andere von Handprogrammierern geliefert werden. Eine gemeinsame OIL-Beschreibung dient als Eingabe für den System-Generator, der das Betriebssystem optimal konfiguriert. Der so angepasste Echtzeitkern wird zusammen mit den anderen Modulen zum endgültigen ausführbaren Programm verbunden, **Bild 4**.

5.1 Konsistenz durch eine gemeinsame OIL-Datenbasis

Zur Erzeugung der textuellen OIL-Datei gibt es verschiedene Möglichkeiten. Zum einen sollten die Eigenschaften der aus dem Simulink-Modell stammenden Betriebssystemobjekte sinnvollerweise direkt auf Modellebene spezifiziert sowie die zugehörigen OIL-Abschnitte automatisch generiert werden können. Zum anderen liefern die meisten OSEK/VDX-Betriebssystem-Anbieter komfortable grafische Tools zur Konfiguration der OIL-Datei. Deren Nutzung ist sicherlich für die handgeschriebenen Programmteile sinnvoll.

Für ein nahtloses Zusammenspiel beider Werkzeuge benutzt TargetLink eine externe OIL-Datei als Datenbasis für die Betriebssystemeinstellungen, aus der importiert und in die exportiert werden kann. Simulink-Komponenten (zum Beispiel Subsysteme) können OSEK/VDX-Objekten (zum Beispiel Tasks) zugewiesen werden, die bereits in der Datenbasis existieren, oder es können neue Einträge angelegt werden. Der Anwender kann frei entscheiden, welches das für ihn geeignetste Tool zum Spezifizieren der Attribute ist.

Die Verwendung der gemeinsamen, externen OIL-Datenbasis gewährleistet Konsistenz – Änderungen, die in einem Tool vorgenommen werden, werden also automatisch vom anderen übernommen.

6 Simulation von OSEK-Anwendungen

Eine der großen Stärken von Simulink ist, dass das Verhalten eines entworfenen Steuergeräts direkt auf dem PC simuliert werden kann. Für den Test des Steuergerätealgorithmus wird häufig ein Streckenmodell benutzt, um den Regelkreis zu schließen, **Bild 5**.

Während der Entwurfsphase wird dieses Modell von Simulink Block für Block in höchster Fließkomma-Genauigkeit simuliert. Die gleiche Anordnung kann in der TargetLink-Umgebung benutzt werden, um in der Implementierungsphase den automatisch generierten Seriencode zu testen und beispielsweise die Quantisierungseffekte einer Festkomma-Implementierung zu untersuchen.

In diesem Simulationsmodus wird anstelle der originalen Simulink-Blöcke ('Controller'-Subsystem aus Bild 5), der generierte C-Code in die Simulation eingebettet (Software-in-the-Loop). Dieser Code wird entweder auf dem Entwicklungs-PC, im Nachfolgenden Host-PC genannt, ausgeführt (Production Code Host Mode) oder als spezielle Option auf einem realen Microcontroller Board (Production Code Target Mode). In beiden Fällen wird das Streckenmodell, das die Teststimuli erzeugt, von Simulink auf dem Host-PC berechnet.

Da in diesen Simulationsmodi der gleiche Code benutzt wird, der nachher auch auf dem Steuergerät läuft, bekommt der Anwender schon in der Simulationsphase exakt die gleichen Berechnungsergebnisse wie in der späteren realen Applikation. Das gilt insbesondere für den 'Production Code Target Mode', da hier der endgültige Seriencode-Compiler zum Übersetzen des C-Programmes benutzt wird und so auch dessen Effekte berücksichtigt werden.

Der in der Simulation verwendete Seriencode enthält im Falle einer OSEK-Anwendung Betriebssystemaufrufe, deren Funktion in der Simulation nachgebildet werden muss. Dazu bietet TargetLink folgende zwei Möglichkeiten.

- Simulation mit spezieller OSEK Nachbildung: TargetLink stellt für die Simulation ein vereinfachtes OSEK-Betriebssystem zur Verfügung, das alle im generierten Seriencode aufgerufenen Betriebssystem-

6 Simulation von OSEK-Anwendungen

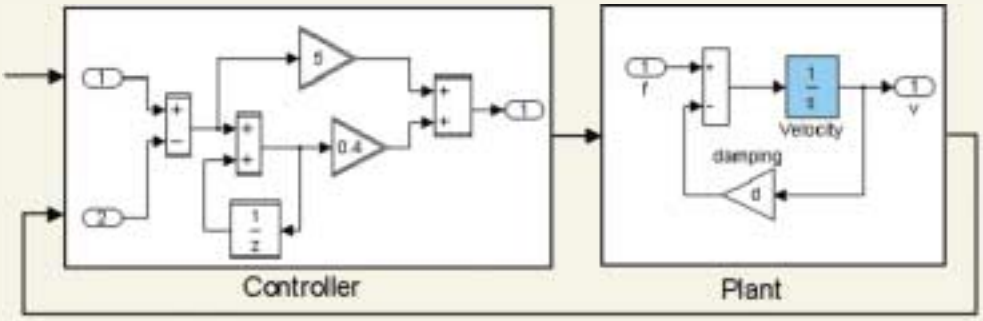


Bild 5: Fließkomma-Simulation

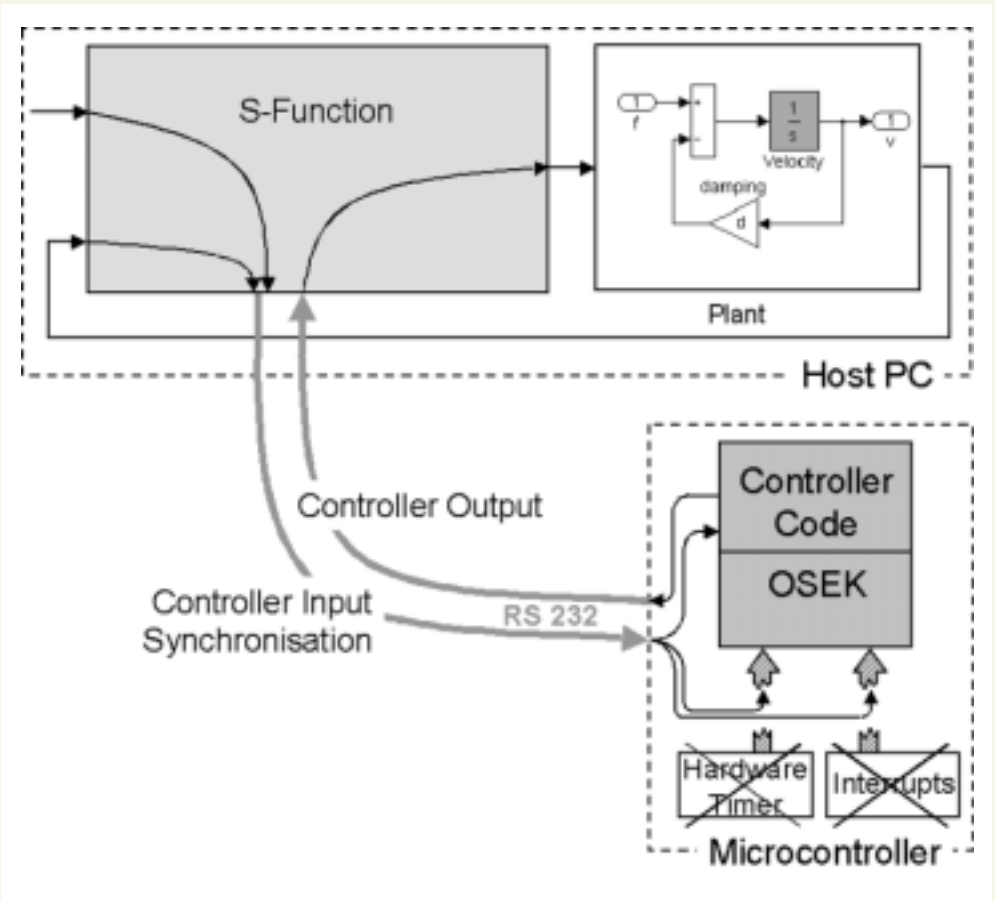


Bild 6: Production Code Target-Simulation mit OSEK/VDX-Betriebssystem

funktionen abbildet. Diese 'OSEK-Nachbildung' ist auf die Anforderungen dieser 'Software-in-the-Loop'-Simulation zugeschnitten und nutzt auf diese Weise die Vereinfachungen aus, die sich aus der speziellen Nicht-Echtzeit-Simulationsumgebung ergeben.

◆ Simulation mit echtem OSEK Betriebssystem: Das Ziel des Entwicklers ist es, schon in der Simulationsphase möglichst viele der später verwendeten Software-Bausteine und deren Zusammenspiel zu testen. Dazu wird in der 'Production Code Target Simulation' das endgültige (kommerzielle) OSEK-Betriebssystem des späteren Steuergerätes verwendet und der generierte Code ('Controller'-Subsystem aus Bild 5) zusammen mit diesem OSEK-Betriebssystem auf einem Mikrocontroller-Board implementiert. Gegenüber der späteren Echtzeitausführung werden für die Simulation Modifikationen vorgenommen.

Da der Fortschritt der Simulation auch von Modellteilen abhängt, die auf dem Host-PC gerechnet werden (Streckenmodell), und daher nicht Echtzeit entspricht, muss die Zielhardware von ihren Echtzeit-Eingängen – Hardware-Timern und Hardware-Interrupts – abgetrennt werden. Diese Stimuli werden vom Host-PC (Simulink) entsprechend der aktuellen Simulationszeit geliefert, **Bild 6**. Dazu wird anstelle des 'Controller'-Subsystems in Simulink ein Programmteil (S-Funktion) eingesetzt, das die Schnittstelle zum Microcontroller-Board bedient.

7 Spezifische Codegenerierung für kommerzielle OSEK/VDX-Betriebssysteme

OSEK/VDX lässt eine ausreichende Flexibilität zu, um die Eigenschaften der verschiedenen Mikrocontroller optimal auszunutzen. Deshalb können bestimmte

Funktionen in verschiedenen OSEK-Betriebssystemen unterschiedliche Benutzerschnittstellen haben. Beispiele hierfür sind:

- API-Funktionen für Counter, welche nicht in OSEK/VDX festgelegt sind
- Unterschiede bei den OIL-Attributen. Viele sind schon im OSEK/VDX-Standard definiert, jedoch können zusätzlich beliebig viele Attribute vom Hersteller definiert werden.

TargetLink benutzt, sofern es nötig ist, auch die vom jeweiligen Betriebssystemhersteller zusätzlich zum Standard angebotenen Betriebssystemfunktionen, um effizienten und lesbaren Code zu erzeugen. Der Anwender erhält die Möglichkeit, auch diese Attribute in der TargetLink-Umgebung einzustellen.

Möchte ein Anwender sein Programm von einem OSEK/VDX-Betriebssystem auf ein anderes übertragen, ersetzt TargetLink automatisch die betriebssystemspezifischen Funktionsaufrufe und OIL-Attribute. Das erhöht die Portierbarkeit und die Wiederverwendbarkeit. Nicht zuletzt führen die Kooperationen zwischen OSEK/VDX-Anbietern und dSPACE zu einem optimalen Zusammenspiel zwischen den einzelnen OSEK/VDX-Betriebssystemen und TargetLink.

8 Zusammenfassung

Eine ständig wachsende Anzahl von Anwendern in automotiven Entwicklungsbereichen nutzt MATLAB/Simulink/Stateflow als Spezifikationsbasis für den gesamten Entwicklungszyklus elektronischer Steuergeräte.

Möglich wird dies auch durch den Einsatz von TargetLink, einem Codegenerator, der aus Simulink-Modellen automatisch seriencodetauglichen C-Code generieren kann. Für die Implementierung der vollständigen Steuergeräte-Software ist der Anschluss an ein Echtzeitbetriebssystem und dabei besonders an den weit verbreiteten Standard OSEK/VDX von großer Bedeutung. TargetLink wird in seiner nächsten Version die Modellierungskonzepte von Simulink noch umfassender, und wenn gewünscht, auch vollautomatisch auf OSEK/VDX-Funktionalität abbilden können. Funktionsentwickler werden weiterhin in ihrer vertrauten Umgebung arbeiten und sich trotzdem die OSEK/VDX-Dienste und Eigenschaften im Modell nutzbar machen können.

Literaturhinweise

[1] Köster, L., Thomsen, T, Stracke, R.: Connecting Simulink to OSEK/VDX: Automatic Code Generation for Real-Time Operating Systems with TargetLink, SAE Technical Paper 2001-01-0024, März 2001
[2] OSEK/VDX Operating System, Version 2.1 revision 1, November 2000
[3] OSEK/VDX System Generation OIL: OSEK Implementation Language, Version 2.2, Juli 2000
[4] OSEK/VDX Communication, Version 2.2.2, Dezember 2000
[5] TargetLink Production Code Generation Guide, dSPACE GmbH, Mai 2000
[6] Simulink User's Guide, The MathWorks, Nattick, MA USA 1999
[7] Hanselmann, H., Kiffmeier, U., Köster, L., Meyer, M.: Automatic Generation of Production Quality Code for ECUs, SAE Technical Paper 1999-01-1168, März 1999