

# Embedded Systeme



## Sonderdruck für Vector Informatik

### *Emulation von CANopen-Steuergeräten unter OSEK/VDX*

## Steuergerätesoftware ohne Hardware testen

**Systeme aus vernetzten Steuergeräten, die über CANopen kommunizieren, werden immer komplizierter. Während die Gesamtentwicklung in der Regel unter hohem Zeitdruck erfolgen muss, steigen gleichzeitig die Anforderungen an die Funktionalität solcher Systeme. Um trotz dieser Herausforderungen das Gesamtsystem zeit- und kostengerecht fertig stellen zu können, ist eine Verifikation des Systems zum frühestmöglichen Zeitpunkt erforderlich oder zumindest sinnvoll. Leider sind jedoch die Prototypen für die Steuergeräte erst zu einem sehr späten Zeitpunkt verfügbar. Selbst wenn der Applikationscode für die Steuergeräte parallel zu deren Hardwareentwicklung geschrieben wird, kann erst nach der Verfügbarkeit der Hardware eine Systemverifikation vorgenommen werden. Es gibt jedoch eine Möglichkeit, die gesamte Steuergerätesoftware gemeinsam mit der Ein- und Ausgangsperipherie und den angeschlossenen Netzwerken zu emulieren. Damit ist in allen Entwicklungsphasen des Projekts eine Verifikation des Gesamtsystems möglich.**

In der Steuergeräteentwicklung wird seit Jahren CANoe/DENoe – ein PC-basiertes Simulationswerkzeug für Netzwerke von Vector Informatik – zur Unterstützung des Entwicklungsprozesses vernetzter Systeme von der Planung bis zur Inbetriebnahme eingesetzt. Zu Beginn des Entwicklungsprozesses entwirft der Netzwerkplaner – in der Regel der Systemhersteller – ein funktionsales Modell der systemweiten Bussysteme. Anhand dieses Modells werden die an der Kommunikation beteiligten Steuergeräte in ihrem Kommunikationsverhalten festgelegt und die dafür erforderlichen Parameter definiert. Das Modell beinhaltet beispielsweise alle verwendeten Botschaften und Signale zusammen mit ihren Parametern wie CAN-Identifizier oder Sendart. An diesem rein virtuellen Netzwerk können bereits wichtige Faktoren wie Busauslastung und Latenzzeiten abgeschätzt werden. Dies ist im ersten Schritt von Bild 1 dargestellt. Eine besondere Eigenschaft des Werkzeugs CANoe/DENoe ist die Kopplung eines simulierten Netzwerks mit dem realen Bus. Damit kann ein Steuergerät, sobald es verfügbar ist, gemeinsam mit dem simulierten Restbus verifiziert werden. Die Struktur dieser Restbussimulation zeigt Bild 1 im zweiten Schritt.

Die Nachbildung der zu simulierenden Netzwerke bei einer Restbussimulation erfolgt innerhalb CANoe/DENoe mit der C-ähnlichen Programmiersprache CAPL. Weiterhin ist es

möglich, DLLs in die Simulation einzubinden, die höhere Protokollschichten wie zum Beispiel CANopen enthalten. Der dritte Schritt (Bild 1 unten) im Entwicklungsablauf ist der Integrationstest beim Systemhersteller. Im Simulationsmodell werden die ursprünglich rein virtuellen Knoten schrittweise durch reale Hardware ersetzt.

Für eine möglichst aussagekräftige Verifikation der zu entwickelnden Steuergeräte ist es notwendig, diese so »real« wie möglich in der Simulationsumgebung abzubilden. Im Idealfall wird der exakt gleiche Applikationscode sowohl für die Simulation als auch für das reale Steuergerät verwendet. Diese Aufgabe wird gelöst, indem die üblicherweise in C geschriebene Steuergerätesoftware in eine DLL umgewandelt wird. Diese DLL wird in CANoe/DENoe den zugehörigen Netzwerke als Verhaltensmodell zugeordnet. Für die Erstellung der DLL steht dem Entwickler eine osCAN.CANoe-Bibliothek zur Verfügung. Diese enthält die Ablaufumgebung für ein OSEK/VDX-konformes Betriebssystem. Zusätzlich steht eine CAN-Treiberschicht zur Verfügung, die über eine CAN-Schnittstelle am PC kommuniziert. Das Betriebssystem osCAN – als Echtzeit-Kernel für über 30 Mikrocontrollerfamilien verfügbar – ist eine weltweit sehr häufig eingesetzte Ablaufumgebung von Steuergeräteapplikationen.

Zu Beginn der Entwicklung muss der Applicationscode für das zu emulierende Steuergerät entworfen werden. Dabei handelt es sich in den meisten Fällen um die eigentliche Funktionalität des Steuergeräts. Für einen Antrieb wären hier zum Beispiel die Regelalgorithmen und die Ansteuerung der Leistungsendstufe zu realisieren. Anschließend wird der CANopen-Source-Code soweit konfiguriert, dass die Anforderungen, die an das Steuergerät gestellt werden, erfüllt sind. Diese Konfiguration umfasst Festlegungen über den Umfang der zur Verfügung gestellten Kommunikationsmechanismen und die genutzten Protokolle. Weiterhin muss der Entwickler auch bestimmen, welche Applikationsdaten das CAN-open-Steuergerät über den Bus zur Verfügung stellt und in welchen CAN-Nachrichten diese Daten übertragen werden sollen. Unter CANopen stehen Applikationsdaten über ein so genanntes Objektverzeichnis zur Verfügung. Dieses Objektverzeichnis ordnet Daten und Funktionen der Applikation eindeutig eine Identifikation zu, die über einen Index (16-Bit-Wert) und einen Sub-Index (8-Bit-Wert) gebildet wird. Soll über den CAN-Bus auf Daten zugegriffen werden, wird dieser Index als »virtuelle« Adresse genutzt. Bei der Entwicklung des Applikationscodes wird auch festgelegt, ob einzelne Aufgaben in separate Betriebssystem-Tasks ausgegliedert werden und welche Betriebssystemmechanismen genutzt werden. Für den CANopen-Source-Code wurde eine solche Aufteilung in einzelne Tasks bereits vorgenommen. Hier ist kein zusätzlicher Eingriff notwendig.

Der Zugriff auf den CAN-Controller wird ebenfalls über die osCAN-CANoe Bibliothek zur Verfügung gestellt. Der CANopen-Source-Code ist bereits für eine große Anzahl von verschiedenen Hardwareplattformen verfügbar. Bei der Verwendung unter CANoe Option CANopen nutzt der Source-Code die gleiche Schnittstelle

zum CAN-Controller wie auf diesen Plattformen. Lediglich die Treiberaufrufe werden auf Funktionen in der osCAN-CANoe-Bibliothek abgebildet. Ist die Applikation erstellt, so wird der Applikationscode gemeinsam mit dem CANopen-Source-Code über die Microsoft-Visual-C++-Entwicklungsumgebung zu einer DLL kompiliert und in CANoe.CANopen eingebunden. Zu dieser DLL wird ebenfalls die osCAN.CANoe-Bibliothek hinzugelinkt. Diese Bibliothek beinhaltet einen vollständigen CAN-Treiber sowie, mit Ausnahme von Teilen der Interrupt-Behandlung, das vollständige Betriebssystem osCAN. Bild 2 zeigt die Struktur der Steuergeräteemulation unter Verwendung des Original-Steuergerätes. Da der Code für das Steuergerät nicht direkt verwendet werden kann, muss die steuergerätespezifische Ein-/Ausgabehardware nachgebildet werden. Das Verhalten der Hardware lässt sich am besten über die C-ähnliche Programmiersprache CAPL nachbilden. Hier können auch komplexere Algorithmen und Zustandsmaschinen hinterlegt werden, mit deren Hilfe eine Annäherung an das Verhalten der realen Hardware leicht möglich ist. Zusätzlich können noch Panels kreiert werden, über die Schalter oder Displays nachgebildet werden. Damit kann die Simulation auch visuell ansprechend gestaltet werden.

Der Entwickler muss alle Zugriffe auf die Hardware, zum Beispiel auf den Interrupt-Controller oder auf I/O-Kanäle, zum CANoe.CANopen-Interface umleiten. Interrupt-Serviceroutinen können jedoch im originalen Steuergerätecode erhalten bleiben und sind mittels einer CAPL-Funktion durch den Entwickler manuell oder durch einen geeigneten CAPL-Algorithmus bedienbar. Wenn es im Applikationscode des Steuergeräts eine Interrupt-Serviceroutine mit dem Namen »ISR\_NewData« gibt, kann dieser Interrupt aus der CAPL-Umgebung heraus mit dem Aufruf OSEK\_Trap

(»ISR\_NewData«) ausgelöst werden. Die Verbindung zwischen der nachgebildeten Ein-/Ausgabehardware und dem Applikationscode erfolgt über »Umgebungsvariablen«, einem in CANoe.CANopen definierten Prozess-Interface. Diese Umgebungsvariablen können Integerwerte oder Strings aufnehmen, auf die dann aus der CAPL-Umgebung oder über Panels zugegriffen werden kann. Für den Zugriff aus dem Applikationscode heraus stehen Funkti-

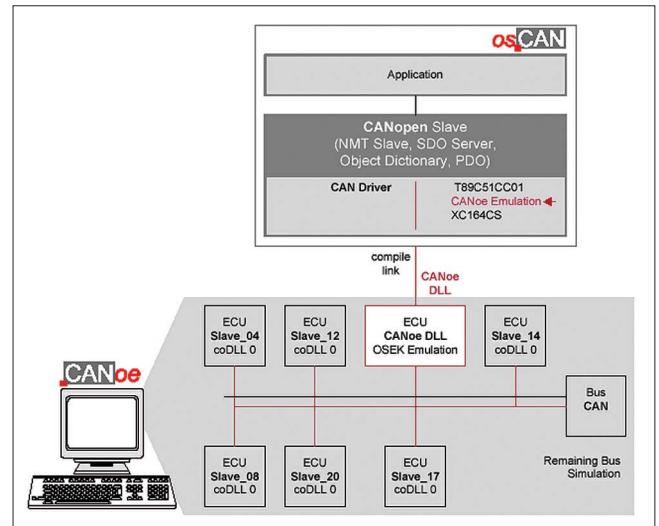


Bild 2. Integration von Steuergerätecode in die CANoe.CANopen-Simulationsumgebung

onen zur Verfügung, die es erlauben, den Wert von Umgebungsvariablen zu lesen beziehungsweise zu schreiben. Überall dort, wo im Applikationscode auf Hardware zugegriffen wird, werden diese Zugriffe auf geeignete Umgebungsvariablen abgebildet. Wenn aus dem Applikationscode heraus eine Leuchtdiode eingeschaltet werden soll, würde das für eine reale Hardware mit folgender Anweisung realisiert werden:

»P3 |= 0x0001;«

Damit wird in der Hardware ein Port-Pin des Port 3 zu »1« gesetzt. In der Emulationsumgebung würde demgegenüber die Funktion: CANoeAPI\_PutValueInt (»LED\_1«, 1); genutzt werden. Dieser Aufruf führt dazu, dass die Umgebungsvariable »LED\_1« den Wert 1 erhält. Diese Umgebungsvariable kann wiederum direkt in einem Panel grafisch dargestellt werden.

Durch einen speziell an CANoe.CANopen angepassten CAN-Treiber steht für den Zugriff auf den CAN-Bus ein Interface zur Verfügung, das direkt mit dem CANopen-Source-Code-API der Vector Informatik genutzt werden kann. Deshalb sind beim Umstieg von der Emulation auf die Zielhardware keine Änderungen für die CAN-Anbindung innerhalb der Steuergeräteapplikation notwendig. Hier muss lediglich die

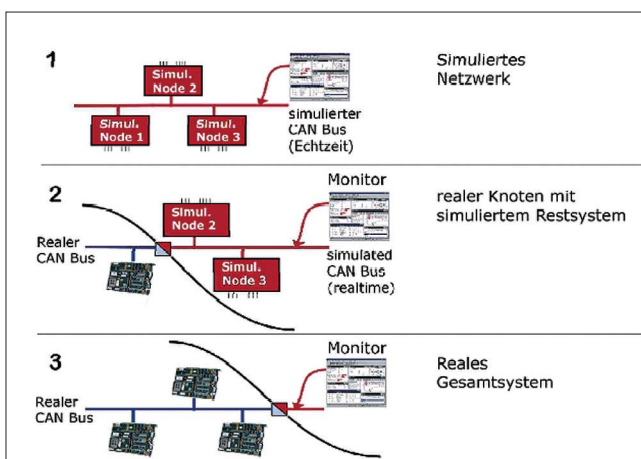


Bild 1. Entwicklungsschritte bei der Simulation eines Netzwerks

Treiberdatei für die gewählte Zielplattform eingesetzt werden. Auch die Behandlung der CAN-Interrupts erfolgt bereits in diesem Treiber und erfordert keine weiteren Maßnahmen von Seiten des Entwicklers. Im Prinzip kann diese Vorgehensweise auf alle zu emulierenden Steuergeräte im System angewendet werden. Für jedes einzelne Steuergerät ist demzufolge eine separate DLL zu erstellen.

Mit dieser Methode wird aus dem Netzwerksimulationswerkzeug CANoe.CANopen ein Emulationswerkzeug. CANoe.CANopen ermöglicht damit nicht nur eine Nachbildung des Netzwerkverhaltens eines oder mehrerer Steuergeräte, sondern darüber hinausgehend auch die Analyse des Ablaufverhaltens der Steuergeräteapplikation und deren funktionalem Verhalten. Task-Wechsel im Betriebssystem können genauso untersucht werden wie die Inter-task-Kommunikation und die Buskommunikation.

Da in einem kompletten System immer mehrere Steuergeräte miteinander interagieren müssen, ist für jedes einzelne Steuergerät eine entsprechende Emulation zu erstellen. Das kann je nach Anzahl der Steuergeräte einen relativ großen Aufwand für den Systementwickler bedeuten. Die Simulationsumgebung lässt sich aber schneller erstellen, wenn für jedes zu simulierende Steuergerät zuerst ein Rahmengerüst generiert wird. Ein solches Rahmengerüst enthält mit dem Netzwerkmanagement (NMT), dem Objektverzeichnis und den fertig konfigurierten Prozessdatenobjekten bereits die Grundfunktionalität eines CAN-

Umgebungsvariablen zur Verfügung.

Die CAPL-Programme und die Panels werden durch das Werkzeug ProCANopen generiert, welches im Lieferumfang von CANoe.CANopen enthalten ist. Natürlich muss auch bei einer automatischen Generierung der Systementwickler eingreifen und festlegen, wie das Steuergerät funktional aussehen soll. Über ProCANopen kann der Systementwickler in einer grafischen Darstellung angeben, welche Steuergeräte später im Gesamtsystem enthalten sein sollen. Die Eigenschaften der einzelnen Steuergeräte in Bezug auf CANopen werden separat durch ein sogenanntes »Elektronisches Datenblatt« festgelegt. Dieses elektronische Datenblatt gibt die Struktur des Objektverzeichnisses vor und definiert wesentliche Geräteeigenschaften. Weiterhin legt der Systementwickler mithilfe von ProCANopen fest, welche Daten zwischen den einzelnen Steuergeräten über CAN ausgetauscht werden sollen und ob dieser Austausch zyklisch oder ereignisorientiert erfolgt.

Sind all diese Festlegungen getroffen, wird per Knopfdruck eine entsprechende Simulationsumgebung erzeugt, die in CANoe.CANopen eingelesen wird. Hier kann man jetzt das Verhalten der simulierten Steuergeräte weiter verfeinern, indem man kleinere Änderungen in den generierten CAPL-Programmen vornimmt. Im weiteren Verlauf werden dann simulierte Steuergeräte nach dem beschriebenen Verfahren durch den realen Applikationscode ersetzt. So gelangt man schrittweise zu einer immer präziseren Simulation.

Die beschriebene Methode erlaubt dem Steuergeräteentwickler das Buskommunikations- und das Applikationsverhalten zu einem sehr frühen Zeitpunkt des Entwicklungsprozesses, ohne das Vorhandensein entsprechender Hardware zu analysieren. Dadurch ergeben sich vielfältige Vorteile:

- Einfaches Debugging der Applikation ist möglich. Wie gewohnt, ermöglicht das Auflaufen des Program-Counters auf einen

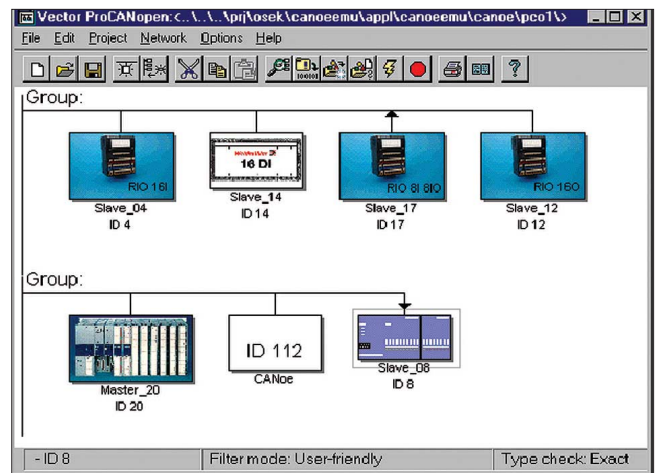


Bild 4. Generieren der Simulationsumgebung über ProCANopen

Breakpoint das detaillierte Analysieren der Systemumgebung, da das komplette System inklusive der Buskommunikation angehalten wird, das heißt, die »globale Zeit« der Emulation wird gestoppt.

- Die Nutzung der weitreichenden CANoe.CANopen-Funktionalitäten wie zum Beispiel das Logging des Datenverkehrs auf dem Bus liefern dem Software-Validierungsprozess wertvolle Informationen.
- Die Analyse des Applikationscodes ist auch ohne den Anteil der Buskommunikation möglich. Als Beispiel sei hier eine Displaysteuerung genannt. Dabei ist der Fokus der Validierung nicht in der Buskommunikation, sondern vielmehr in der Behandlung der Display-I/Os zu sehen.
- Für die Softwarevalidierung stehen den Entwicklungsteams viele Werkzeuge unter Microsoft-Visual-C++ zur Verfügung.

Der Test von Steuergerätesoftware ist im Verbund eines virtuellen oder realen CAN-Netzwerks auch ohne das Vorhandensein von Hardware möglich. Die osCAN-CANoe-Bibliothek bietet dem Steuergeräte-Softwareentwickler die Möglichkeit, mit dem Software-Tool CANoe.CANopen die eigene Applikation vorab zu testen. Durch die Implementierung des OSEK/VDX-konformen Betriebssystems osCAN von Vector Informatik sowie eines auf CANoe.CANopen basierenden CAN-Treibers kann sich der Entwickler ganz auf den Test des selbst erstellten Applikationscodes konzentrieren. Vector Informatik bietet darüber hinaus auch eine Integration der höheren Netzwerkprotokolle wie zum Beispiel das Netzwerkmanagement an, sodass der Entwickler nur das API des entsprechenden Protokolls zu bedienen hat. (Mirko Tischer, Vector Informatik/pa)

**Info:** Vector Informatik

Tel.: 0711/806700, [www.vector-informatik.de](http://www.vector-informatik.de)

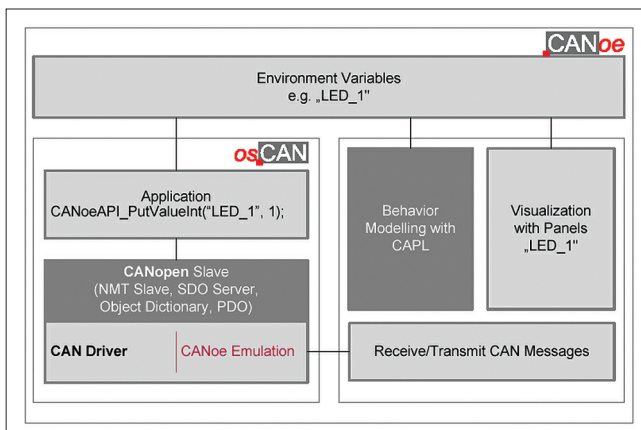


Bild 3. Abbildung von realer Hardware auf Umgebungsvariablen

open-Steuergeräts. Hinter dieser Lösung steckt technisch eine generische Node-Layer-DLL, die CANopen-Verhalten implementiert und ein CAPL-Programm, das diese Node-Layer-DLL konfiguriert und auch kontrolliert. Weiterhin steht automatisch auch ein Panel zur Visualisierung der Prozessdaten des Steuergeräts über