Fahrzeuganbindungen durch Standard-IT-Verfahren

Dipl. Inform. Oliver Hartkopp

Fahrzeugsystemelektronik
Konzernforschung der Volkswagen AG
Brieffach 1776
38436 Wolfsburg
oliver.hartkopp@volkswagen.de

Abstract: Der folgende Text beschreibt im ersten Teil die bisher bei PC-Systemen eingesetzten Verfahren zum Zugriff auf Fahrzeugnetzwerke am Beispiel des Controller Area Network ‚CAN' und deren Nachteile in Bezug auf übliche informationstechnische Modelle und Anforderungen. Im zweiten Teil wird gezeigt, wie auf ein CAN-Fahrzeugnetzwerk mit Verfahren aus der Standard-IT, analog der etablierten Internet-Kommunikation, zugegriffen werden kann und welche technischen und finanziellen Vorteile sich daraus für den Programmierer und den Anwender ergeben können.

1 Einleitung

Die Vernetzung von Informatikern und Ingenieuren scheitert häufig an der Verschiedenartigkeit des technischen Hintergrundes und den selbst gesetzten Prioritäten zur Erreichung eines funktionalen Zieles. So hat sich die automobiltechnisch geprägte Informationstechnik seit der Einführung des Motorsteuergerätes in den 80er Jahren des letzen Jahrhunderts losgelöst von der Standard-Informationstechnologie entwickelt. Als in den 90er Jahren in den eingebetteten Systemen der Automotive-IT die ersten Betriebssysteme eingeführt wurden, waren ausgereifte Unix-Betriebssysteme bereits seit 20 Jahren erfolgreich im Einsatz.

Spätestens seit der Einführung von Steuergeräten, die durch Software-Updates im Fahrzeug modifiziert werden können, existiert eine Verbindung von PC-Systemen (Standard-IT) mit vernetzten Fahrzeugsystemen (Automotive-IT). Obgleich die gestiegene Komplexität, der hohe Vernetzungsgrad und die hohen evolutionsbedingten Entwicklungskosten an dieser Schnittstelle den Einsatz etablierter und getesteter Verfahren und Standards (Schichtenmodell) eigentlich erzwingen, gibt es auf diesem Gebiet nur wenig fachübergreifende Zusammenarbeit.

Das im Folgenden beschriebene „Low Level CAN Framework" realisiert eine nahtlose Integration des Fahrzeugnetzwerkes analog zu etablierten Netzwerkprotokollen in Betriebssysteme wie Linux™ oder Windows™ und stellt somit eine interdisziplinäre Brücke zwischen den Domänen der Standard-IT und der Automotive-IT dar.

2 PC-Anbindungen von Fahrzeugbussen (Stand der Technik)

Zur Anbindung eines Fahrzeugbusses muss im Falle des CAN-Busses ein CAN-Controller-Chip an das PC-System angeschlossen werden. Um auf diesen CAN-Controller mit einer Anwendung zugreifen zu können, wird für das jeweilige Betriebssystem eine Treiber-Software benötigt, den so genannten CAN-Treiber.

Ein CAN-Treiber ist nach dem Stand der Technik als Character-Device, d.h. als „Zeichengerät“ realisiert. Aus Anwendungssicht erfolgt der Zugriff auf den CAN-Bus daher vergleichbar mit der Art des Zugriffes auf eine serielle Schnittstelle.

Eine solche Realisierung hat jedoch für den Anwender gravierende Nachteile:

- Es kann nur eine Anwendung zu einer Zeit auf den CAN-Bus zugreifen.
- Anwenderprotokolle müssen außerhalb des Betriebssystems realisiert werden.
- Es gibt keinen einheitlichen Treiber-Standard für CAN-Controller wie beispielsweise bei Netzwerkkarten.

Ein CAN-Transportprotokoll (z.B. nach ISO 15765-2) muss in der Folge nach dem Stand der Technik in der Anwendungsschicht realisiert werden (siehe Abbildung 1). In diesem so genannten Userspace können die geforderten Protokollantwortzeiten jedoch häufig nicht garantiert bzw. realisiert werden. Bei hoher Systembelastung (z.B. beim Starten von Programmen oder bei Festplattenaktivität) kann daher eine Kommunikation von Seiten des CAN-Steuergerätes abgebrochen werden, weil der PC nicht im geforderten Zeitfenster geantwortet hat.

Um solche Timing-Probleme bei Transportprotokollen zu umgehen, wird stellenweise mit aufwändigen Echtzeitbetriebssystemen gearbeitet, die der *Anwendungsschicht* garantierte Zeitscheiben für die geforderten Antwortzeiten zusichert. Das ist etwa so, als würden Autos mit einem Metronom ausgeliefert werden, zu deren Takt der Anwender bzw. Fahrer dann beim Abbiegen die Blinkleuchte ein- und ausschaltet. In beiden Fällen werden Aufgaben an den Anwender delegiert, die das System selbständig leisten sollte.

Ein weiteres Problem ergibt sich aus der Tatsache, dass nur genau eine Anwendung auf den CAN-Bus zu einer Zeit zugreifen kann. Über Ethernet und Internetprotokoll können beispielsweise mehrere verschiedene Anwendungen gleichzeitig auf das Netzwerk zugreifen. Die aktuelle Realisierungsart für CAN-Treiber zwingt den Anwendungsprogrammierer jedoch dazu, genau *eine* monolithische CAN-Anwendung zu realisieren. Aus diesem Grund sind in bekannten CAN-Werkzeugen (z.B. CANoe der Firma Vector-Informatik) die Simulations-, Tracing-, Statistik- und Transportprotokollfunktionalitäten in ein einziges Programm integriert. Und das obwohl das Tracing von CAN-Botschaften mit der Simulation von Steuergeräten weder logisch noch funktional zusammenhängt.

Abschließend sei noch angemerkt, dass es zu jeder erhältlichen CAN-Hardware eigene Treiber mit jeweils unterschiedlichen Schnittstellen zur Anwendung gibt. D.h. im Speziellen, dass eine Anwendung, die auf einen CAN-Treiber von Hersteller A aufsetzt, nur mit erheblichem Aufwand auf eine andere CAN-Hardware von Hersteller B angepasst werden kann.

3 Das Controller Area Network wird zum PC-Netzwerk

Die bezeichneten Nachteile des bisherigen Standes der Technik führen zu der Fragestellung, wie eine Anbindung des CAN-Busses mit Verfahren aus der Standard-IT sinnvoll zu realisieren ist.

Anforderungen aus Anwendungssicht:

- Jede CAN-Anwendung kann gleichzeitig auf einen oder mehrere CAN-Busse lesend und schreibend zugreifen.
- Es kann mehrere solcher CAN-Anwendungen zu einer Zeit geben.
- Es gibt keine systembedingten Abhängigkeiten zwischen CAN-Anwendungen wie z.B. Locking-Verfahren beim CAN-Buszugriff, etc.
- CAN-Protokolle mit restriktiven Antwortzeiten sollen im Betriebssystem realisiert sein.
- Es sind standardisierte Schnittstellen zum Betriebssystem analog existierender Netzwerkimplementierungen zu verwenden.

Aus diesen Anforderungen ergibt sich, dass der CAN nicht in Form eines „Zeichentreibers“ sondern als Netzwerk realisiert werden muss. Die Anwendungsschnittstelle wird als eine Socket-Schnittstelle realisiert und in ihrer Verwendung an die für die Internet-Kommunikation bekannten TCP/IP-Sockets angelehnt. Für die Anbindung von CAN-Controllern wird das Treibermodell des Netzwerk-Treibers gewählt, das auch bei Ethernetkarten verwendet wird.

Zur Realisierung des so genannten „Low Level CAN Framework“ Konzeptes wurde im Betriebssystem neben der Protokollfamilie für das Internetprotokoll (PF_INET) eine neue Protokollfamilie PF_CAN etabliert, die den CAN-spezifischen Anforderungen und Adressierungsarten Rechnung trägt.

Innerhalb der Protokollfamilie PF_CAN können auf diese Weise im Betriebssystemkontext unter anderem CAN-Transportprotokolle wie ISO 15765-2 realisiert werden.

Eine Besonderheit in der Protokollfamilie PF_CAN stellt der so genannte RX-Dispatcher dar: Durch die Art der inhaltsbezogenen Adressierung der CAN-Frames kann es mehrere Interessenten an einer empfangenen CAN-ID geben. Durch interne Funktionen können sich die Protokollmodule beim PF_CAN-Rahmenmodul für ein oder mehrere CAN-IDs von definierten CAN-Netzwerkgeräten registrieren, die ihnen beim Empfang automatisch zugestellt werden. Das PF_CAN-Rahmenmodul sorgt beim Senden auf den CAN-Bus auch für ein lokales Echo (‚local loopback') der zu versendenden CAN-Botschaften, damit für alle Applikationen auf einem System die gleichen Informationen verfügbar sind.

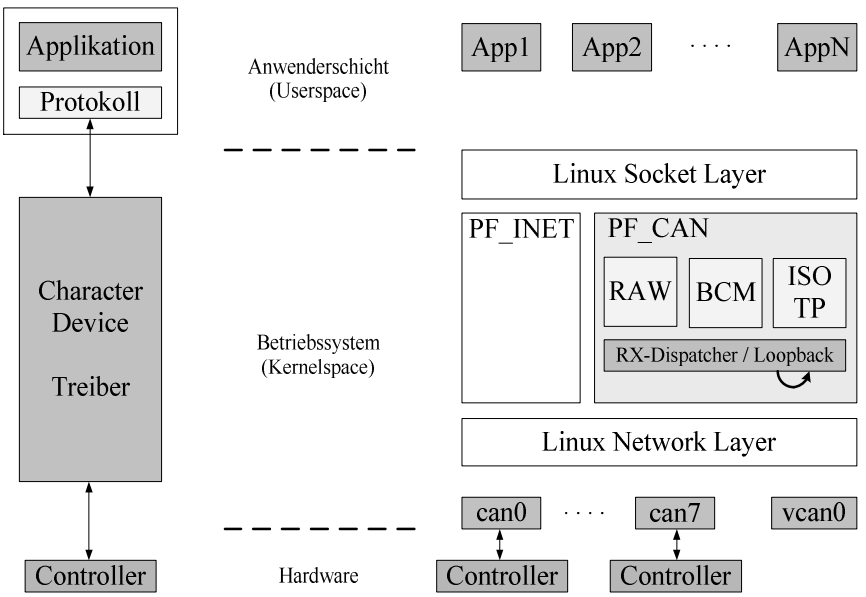

Abbildung 1: Der CAN als Zeichentreiber und als Netzwerkrealisierung

Die Realisierung des CAN-Treibers als Netzwerktreiber unter Linux™ hat sich zudem als sehr viel einfacher herausgestellt, als eine Realisierung als Zeichentreiber. Ausgehend von einer etablierten Betriebssystemschnittstelle für Netzwerkkarten musste sich der Treiberentwickler z.B. mit Fragestellungen wie der Pufferung von CAN-Botschaften und Vergabe von Zeitstempeln (Timestamps) nicht auseinandersetzen, weil solche Techniken bereits im Linux Network Layer vorhanden sind.

Dieses Verfahren, den CAN-Bus als Netzwerk in etablierte Betriebssysteme zu integrieren, wurde beispielsweise auch erfolgreich auf Microsoft WindowsXP™ übertragen, wobei der CAN-Treiber als Miniport-Netzwerktreiber realisiert wurde. Die fahrzeugspezifischen Protokolle werden dabei als „Native Media Aware Protokoll" in das NDIS-Treiberinterface eingebunden.

4 Zusammenfassung und Ausblick

Die Realisierung des Controller Area Network als Netzwerk im Betriebssystem und der Zugriff auf Fahrzeugsysteme über die standardisierte Socket-Schnittstelle stellt eine grundlegende technische Neuheit für Feldbusse im industriellen und automobiltechnischen Umfeld dar. Durch den Einsatz von IT-Standards verringern sich die Einarbeitungszeit von Entwicklern und die möglichen Fehlerquellen erheblich. Es ist unter Verwendung von etablierten System-Aufrufen ebenso einfach eine Punkt-zu-Punkt-Verbindung über das Internet herzustellen, wie zu Diagnosezwecken zu einem Motorsteuergerät. Insbesondere bei der Schnittstelle zu IT-Anwendungen bei prototypischen Fahrzeugen, im Car-to-Car-Bereich (mit Wireless LAN), im Bereich (telemetrischer) Diagnoseanwendungen und beim Parallelisieren von Softwareupdates im Fahrzeug wird das besondere Potenzial dieses neuen Verfahrens sichtbar.

Das seit dem Jahre 2003 von der Konzernforschung der Volkswagen AG entwickelte „Low Level CAN Framework“ wurde im Dezember 2005 unter der GNU Public License als Open Source freigegeben, um eine Integration in den Standard Linux Kernel zu ermöglichen. Seit Februar 2006 werden diese Quelltexte auf einem öffentlich zugänglichen Open Source Server mit Anregungen aus einer zunehmenden Socket-CAN Comunity zusammen mit Volkswagen weiterentwickelt.

Literaturverzeichnis

[Ha06] Oliver Hartkopp: Low Level CAN Framework, Application Programmers Interface Version 1.0.0, http://www.llcf.de/downloads/llcf-api-2006-02-20.pdf, 2006.

[Lü04] Dr. Andreas Lübke, Car-to-Car-Kommunikation – Technologische Herausforderungen, VDE, http://www.network-on-wheels.de/downloads/VDE2004_Luebke_Paper.pdf, 2004

[Be06] Berlios Open Source Mediator: Open Source Projekt-Homepage des Low Level CAN Frameworks: http://developer.berlios.de/projects/socketcan

[CRK05] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, Linux Device Drivers, O'Reilly Media, ISBN 0596005903, 3. Auflage, 2005

[RFC03] Request for Comment RFC 3493: Basic Socket Interface Extensions for IPv6, February 2003, http://www.ietf.org/rfc/rfc3493.txt

[La00] Wolfhard Lawrenz, Controller Area Network : CAN ; Grundlagen und Praxis, 4. überarbeitete Auflage - Heidelberg : Hüthig, 2000

[CiA06] CAN in Automation, http://www.can-cia.org