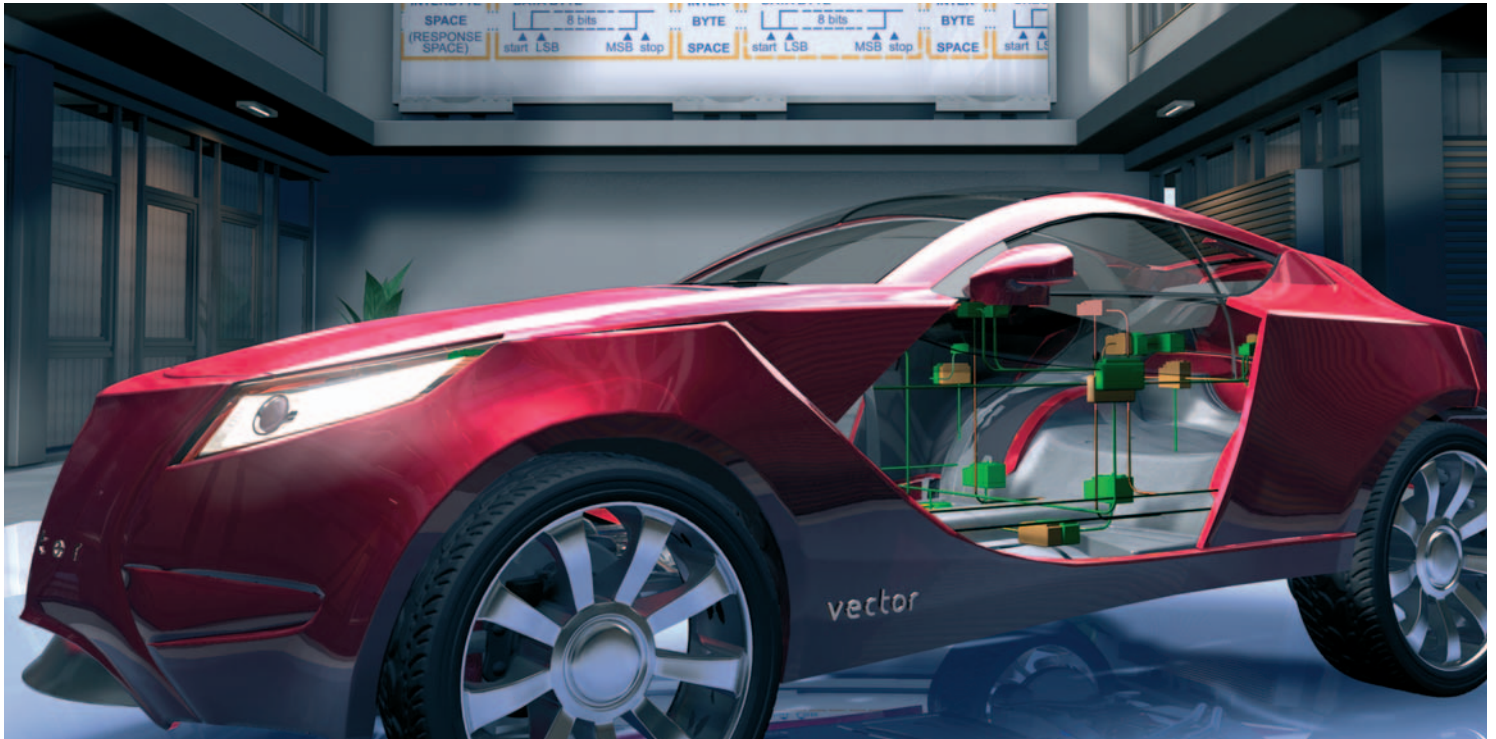


Alternative Testverfahren für LIN-Steuergeräte

Perspektiven für den LIN-Master-Conformance-Test



Der Test des LIN-Masters spielt bei der Verifikation eines LIN-Netzwerkes eine zentrale Rolle, da dieser die LIN-Kommunikation maßgeblich bestimmt. Für die methodische Umsetzung nutzt man in der Praxis den sogenannten Gray-Box-Test, bei dem für die Teststimulation und -verifikation sowohl externe Schnittstellen herangezogen werden als auch Eingriffe in die Interna des zu testenden Geräts notwendig sind. Weil diese Vorgehensweise mit einigen Nachteilen verbunden ist, wurde der LIN-Black-Box-Master-Conformance-Test konzipiert. ■ Andreas Pick



Andreas Pick
ist Produktmanager
für LIN-Embedded-Softwarekomponenten
bei Vector Informatik in Stuttgart
T +49/711/80670-360
andreas.pick@vector-informatik.de

Bei Gray-Box-Tests greift man mithilfe einer Testapplikation und einem zugehörigen Test LIN Description File (LDF) direkt auf den Master-Treiber zu, wobei für Stimulation und Ablaufsteuerung der Tester verantwortlich ist. Damit erreicht man zwar eine volle Testabdeckung, muss aber auch den wesentlichen Nachteil in Kauf nehmen, dass sich der übliche V-Modell-Entwicklungsprozess nicht einhalten lässt. Dies liegt insbesondere daran, dass die Testapplikation nicht mit der eigentlichen Steuergerätesoftware mit ihren

spezifischen Aufgaben identisch ist und für den Test jeweils speziell aufgesetzt werden muss. Eine Verifikation der Master-Konformität ist folglich nur am Anfang des Entwicklungsprozesses möglich. Sobald die produktive Datenbasis (LDF) und Applikation auf dem Master-Steuergerät laufen, sind weitere Verifikationen während der Entwicklung nicht mehr möglich.

Auch die Testvorbereitung ist aufgrund der Konfiguration des Treibers mit dem Test-LDF und der jeweiligen Neukompilierung mit erheblichem Aufwand verbunden. Hierbei be-

nötigt der Fahrzeughersteller die Unterstützung der Zulieferer hinsichtlich Compiler und Download-Tools. Der Vorteil von Black-Box-Tests liegt darin, dass für sämtliche Testaktivitäten ausschließlich die vorhandenen externen Schnittstellen zur Anwendung kommen.

Conformance-Tests auf Slave-Seite problemlos als Black-Box-Tests umsetzen

Da LIN-Netze nach dem Master-Slave-Prinzip arbeiten, lassen sich die vom LIN-Konsortium spezifizierten Conformance-Tests auf der Seite der Slaves relativ problemlos als Black-Box-Tests realisieren. Der Slave erhält vom Master die Testanfrage bzw. Stimulation über die Standard-LIN-Schnittstelle und schickt ebenso über diese die Antwort zurück. Ein ähnliches Black-Box-Verfahren für den Master-Conformance-Test würde während des gesamten Entwicklungsprozesses unabhängige Tests mit minimalem Konfigurationsaufwand sowohl beim OEM als auch beim Zulieferer ermöglichen.

In der aktuellen Version des Entwicklungs- und Test-Tools CANOE/LIN 7.0 sind die Slave-Conformance-Tests nahezu vollständig als Black-Box-Test implementiert, während man für den Master-Conformance-Test nach wie vor das Gray-Box-Verfahren anwendet. Dazu liefert Vector ein spezielles Test-LDF zusammen mit einer Testapplikation aus, die auf das Master-Steuergerät zu laden sind. Zur Verifikation nutzt man den LIN-Bus.

Die wesentliche Anforderung des Testverfahrens besteht darin, einerseits die Stimulation unter gewissen Rahmenbedingungen zu leisten und andererseits die regulären applikativen Funktionen bereitzustellen bzw. nicht zu beeinträchtigen, zum Beispiel die einer Schei-

Testkommando	Parameter	Test-ID
Test LogIn	keine	0x00
Test Presence / Idle	keine	0x01
Test Setup / Trigger	T _{base} Cycles, Reset / Init indication	0x02
Load Parameters	Offset, D1, D2, D3	0x03
Read Result	Offset, D1, D2, D3, D4	0x04
Set schedule table	Table index, Slot index, T _{base} Cycles	0x05
Read Status Word	PID / keine	0x06
Send Config Command	RCSID, RCNAD, RCPCI	0x07
Request Sleep Mode	Sleep cyc., Schedule tab, wakeup Del.	0x08
Set signal	Signal handle, Signal value	0x09
Read signal	Signal handle	0x0A

Abb. 1: Übersicht aller elf Testkommandos, die für eine vollständige Ausführung des aktuellen LIN-2.0-Conformance-Tests erforderlich sind

benwischer-Steuerung. Für den Black-Box-Master-Conformance-Test muss man die Testkommunikation von der üblichen Kommunikation mit den bekannten Dateninhalten, den verschiedenen Schedule Tables anhand eines eindeutigen Kriteriums unterscheiden können. Dazu wurden die existierenden Protokolle der Transport- und Diagnose-Schicht einschließlich des Master-Requests und Response-Mechanismus' für die Testimplementierung auf Single-Frame-Basis weiterverwendet. Als spezieller Diagnoseservice für die eindeutige Unterscheidung der Testkommunikation bietet sich der bislang nicht belegte Service Identifier 0xB8 der LIN 2.x „Node Configuration and Identification Specification“ an.

Die Prototyp-Implementierung umfasst momentan elf Services, die mit ihren jeweiligen Parametern für die verschiedenen Phasen wie Initialisierung oder Verifikation verantwortlich sind und bestimmte Aktionen im Master

auslösen (Abbildung 1). Damit ist es möglich, die Spezifikationen des aktuellen LIN-2.0-Conformance-Tests vollständig zu erfüllen. Nach einem erfolgreichen Test-Login und der Testinitialisierung ist das System bereit für die Stimulation. Bei der eigentlichen Testausführung schaltet man dann aus dem Test Communication Schedule Table zum Beispiel in ein spezielles Schedule Table für den Schedule-Table-Test und springt wieder in einen Master Request Slave Response Table zurück, um die Ergebnisse aus dem Master abzuholen.

Während normalerweise der Master die Kommunikation mit den Slaves dominiert, verhält sich die Testkommunikation genau umgekehrt. Der Slave schickt in der Rolle des Testers ein Testkommando an den Master, auf das dieser nun positiv oder negativ reagiert. Die Inhalte des Testprotokolls sind in Anlehnung an das Diagnoseprotokoll-Format und den Service 0xB8 folgendermaßen aufgebaut (Abbildung 2): Das im Slave Response untergebrachte Testkommando besteht am Anfang zunächst aus ISO-Transportprotokoll-Adressierungs- und Diagnose-RSID. Danach folgen der Test Identifier (TID) und vier mögliche Bytes Nutzlast. Die Test-ID referenziert eindeutig ein spezielles Testkommando, das je nach Funktion mit bis zu 4 Bytes Parameterinformationen ergänzt wird. Der Master antwortet auf das Testkommando positiv oder negativ mit einer Master Confir- >

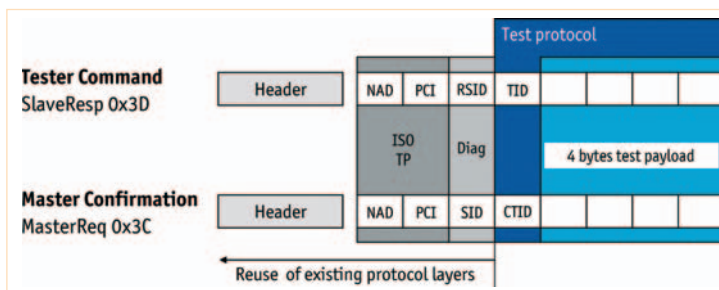


Abb. 2: Format der Testkommandos mit Test ID (TID) und Testparametern (Payload): ISO-Transport- und Diagnoseprotokoll werden auch für die Testkommunikation eingesetzt.

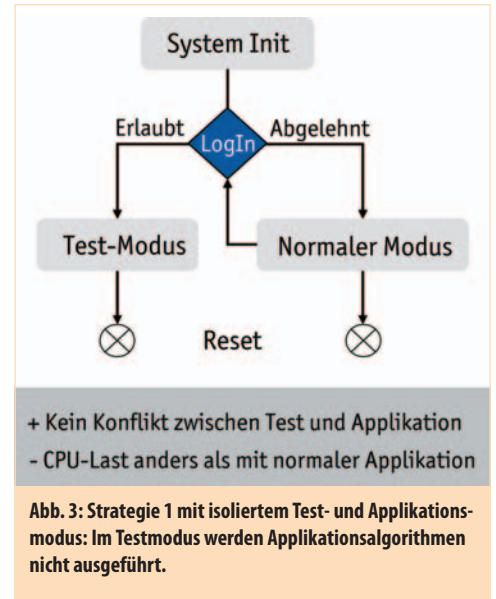
mation, wobei sich hier wieder die aus der Diagnose bekannten Werte anbieten, TID+0x40 für den „Positive Confirmation Test Identifier“ (CTID) sowie 0x7F für den „Negative Confirmation Test Identifier“.

Identischer Code für Test und Produktion verhindert Bruch im V-Modell

Der um diese Testschnittstelle erweiterte LIN-Master-Treiber bietet etliche Vorteile: Während der Tests und der Entwicklung nutzt man eine identische Treibersoftware. Dadurch lässt sich der Treiber stets mit dem produktiven LDF konfigurieren – das gleiche LDF wird also für Test und Entwicklung verwendet. Der gravierende Bruch im V-Modell zwischen Integration und Verifikation entfällt. Die Erweiterungen schränken den existierenden LIN-Treiber, so wie er heute existiert, nicht ein. Schließlich lässt sich der Testcode, falls nötig, über den Präprozessor entfernen.

Der entscheidende Vorteil des aufgezeigten Weges liegt darin, dass man für alle Tests keine zusätzlichen Schnittstellen benötigt und immer die exakt gleichen Bedingungen vorliegen. Die alternative Nutzung der CAN-Schnittstelle hat den Nachteil, dass diese einerseits nicht immer vorhanden ist und andererseits jeweils projektspezifisch konfiguriert werden müsste. Da es unumgänglich ist, jeweils einen Test-Login und eine Testinitialisierung zu starten, hat sich allerdings gezeigt, dass ein völlig unabhängiges Nebeneinander von Testdurchführung auf der einen Seite und Applikations-Algorithmen auf der anderen Seite, nicht realisierbar ist. Die Applikation muss letztendlich im Testmodus also immer berücksichtigt werden.

Zur Interaktion zwischen Applikation und Treiber für Testmodus stehen zwei Strategien zur Verfügung (Abbildungen 3 und 4). Alternativ zu einem Modell, das die Applikation klar darüber informiert, ob ein Test durchgeführt



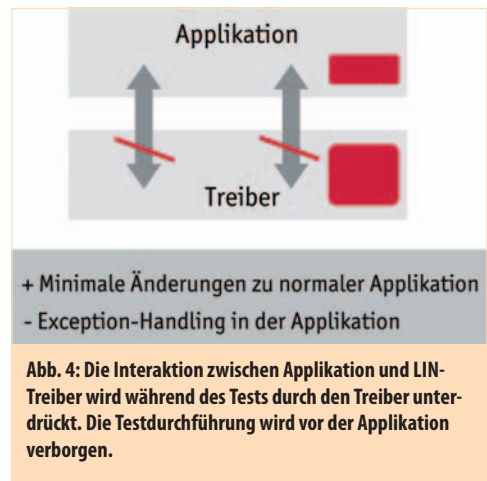
wird oder nicht, kann man diese Information vor der Applikation so weit wie möglich verschleiern.

Standard-Treiber muss um zusätzlichen Code erweitert werden

Welcher Weg zu beschreiten ist und welche Strategie sich mit ihren jeweiligen Vor- und Nachteilen als geeigneter herausstellt, hängt letztlich von den Wünschen von OEMs und Lieferanten ab. In jedem Fall müsste der existierende Treiber, so wie er heute standardisiert ist, um eine gewisse Test-Server-Fähigkeit erweitert werden. Der dafür erforderliche Zusatzcode liegt schätzungsweise bei zusätzlichen 20 bis 30 Prozent, bezogen auf den aktuellen Treiberumfang, was für Master-Steuergeräte in der Regel kein Problem darstellt.

Weiterhin ist das Konzept nicht nur grundsätzlich erweiterbar in Richtung Applikationstests, sondern auch standardisierbar. Denn einerseits basiert es auf dem existierenden LDF samt passendem Entwicklungsprozess, andererseits sind die Kommandos offen gelegt und können ergänzt werden. Schließlich eröffnet eine standardisierte Testschnittstelle im LIN-Master neue Möglichkeiten für Simulations- und Test-Tools, was deutlich effizienter und anwendergerechter ist als heute die Verifikation des LIN-Masters zu realisieren. ■

Dieser Beitrag als PDF und weiterführende Informationen (ähnliche Beiträge, technische Daten, Direktlinks zum Hersteller etc.) sind online verfügbar auf www.EuE24.net



more @ click **EE068013**