
# Das universelle Gateway-Steuergerät

## Flexible Post-Build-Konfiguration von AUTOSAR-Gateways

**Das Post-Build-Prinzip ermöglicht eine nachträgliche Konfiguration von Gateway-Steuergeräten selbst in späten Entwicklungsphasen während der Steuergeräteintegration oder sogar im Feld. Das Ergebnis sind universell einsetzbare Steuergeräte. Basierend auf dem AUTOSAR-Standard, kann mit dem hier vorgestellten Verfahren die Gateway-Funktionalität im fertigen Steuergerät an neue Anforderungen angepasst werden.**

Von Hartmut Hörner

Eingebettete Systeme lassen sich zu verschiedenen Zeitpunkten konfigurieren. Bevor der Quell-Code den Build-Prozess durchläuft, findet die so genannte Pre-Compile-Konfiguration statt. Sie ermöglicht die effiziente Realisierung von Konfigurationen wie beispielsweise Varianten durch Makros oder C-Präprozessorschalter. Die Link-Time-Konfiguration kommt typischerweise zum Einsatz, um eine Bibliothek zu erzeugen und diese mit im ROM gespeicherten Konstanten zu verbinden (linken). Des weiteren gibt es die Möglichkeit zur Konfiguration während der Laufzeit des Steuergeräts (Run-Time-Konfiguration), etwa durch Kalibrierung oder Diagnosekommandos. In solchen Fällen müssen die Konfigurationsparameter im RAM abgelegt sein. Im Gegensatz zu den zuvor genannten Konfigurationsarten findet die Post-Build-Konfiguration im bereits fertig gebauten Steuergerät statt, indem man die Konfigurationsdaten über einen Flash-Bootloader ins Steuergerät lädt **Bild 1**. Der AUTOSAR-Standard definiert drei so genannte Konfigurations-Konformitäts-Klassen (Configuration Conformance Classes, CCC), die diese unterschiedlichen Konfigurationszeitpunkte abdecken: CCC0 bedeutet Pre-Compile-Konfiguration, CCC1 Link-Time-Konfiguration und CCC2 Post-Build-Konfiguration.

Welche Konfigurationsmöglichkeit für ein bestimmtes Basis-Software-Modul zur Verfügung steht, hängt vom Charakter des Moduls ab. Das RTE (Runtime Environment) unterstützt CCC0, weil es eng mit den Applikationen verknüpft ist und aus vollständig generiertem Code besteht. Auch das Betriebssystem wird nur zum Pre-Compile-Zeitpunkt konfiguriert. **Bild 2** zeigt im AUTOSAR-Schichtenmodell, welche Konfigurationsmöglichkeiten für die Module zur CAN-Kommunikation bestehen.

Die zum Kommunikations-Stack gehörenden Module unterhalb des RTE unterstützen zum großen Teil die Post-Build-Konfiguration gemäß CCC2. Trotzdem haben diese Module einige Pre-Compile-Parameter wie beispielsweise die Anzahl der Kanäle, die Verwendung von Datenpuffern (Queues) und die Aktivierung von Debug-Modi. Diese Einstellungen müssen der Bibliothek zum Zeitpunkt der Erzeugung bekannt sein. Während des Entwurfs eines Steuergerätes muss ein sinnvolles Konzept entwickelt werden, damit die Post-Build-Fähigkeiten benachbarter Module zueinander passen. So ist es weniger sinnvoll, für ein einzelnes Modul wie etwa den PDU-Router (PDU-R) eine Post-Build-Fähigkeit vorzusehen, für alle anderen Module aber nur Pre-Compile-Konfigurationen zuzulassen.

Ändert sich die Funktionalität einzelner Steuergeräte z.B. im Rahmen einer Modellpflege, ist vielfach auch eine Änderung in der Kommunikations-Matrix eines oder mehrerer Netzwerke notwendig. Hier setzt der Post-Build-Ansatz ein und ermöglicht bei allen Steuergeräten des betroffenen Netzwerkes die einfache Anpassung der für die Kommunikation zuständigen Basis-Software. Ein erneutes Compilieren und Linken des Codes entfällt. Ist ein Gateway-Steuergerät post-build-fähig ausgelegt, passt es leichter

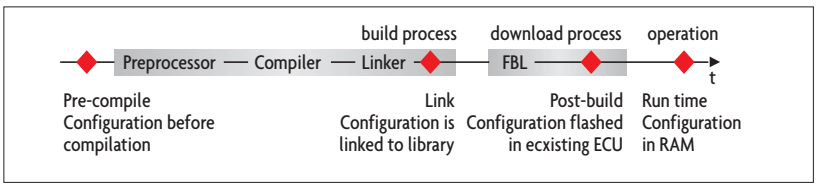


**Bild 1. In der Steuergeräte-Entwicklung gibt es vier unterschiedliche Konfigurationskonzepte.**

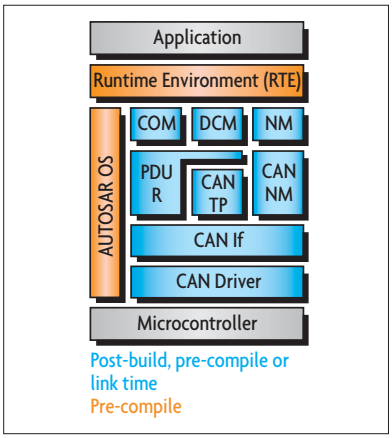


**Bild 2. Unterhalb der RTE sind viele Basis-Software-Module grundsätzlich zum Post-Build-Zeitpunkt konfigurierbar.**

als Standardbauteil in verschiedene Baureihen. Gateway-Steuergeräte erledigen viele ihrer Aufgaben über die Kommunikations-Basis-Software. Eine Modellpflege besteht oft nur aus neuen oder geänderten Routing-Beziehungen, für die lediglich ein Umkonfigurieren der Basis-Software nötig ist.

Die Hauptmotivation für den Einsatz einer Post-Build-Konfiguration ist die Tatsache, dass kein neuer Build-Prozess benötigt wird und damit die Konfiguration in späten Entwicklungsphasen während der Steuergeräte-Integration oder gar im Feld durchgeführt werden kann. Insbesondere für Gateway-Steuergeräte ist dieser Ansatz sehr interessant.

## Das Gateway-Steuergerät als flexible Schaltzentrale

Die Hauptfunktion eines Gateway-Steuergerätes ist die Verteilung der Kommunikationsdaten zwischen den einzelnen Netzwerken eines Fahrzeugs. Gemäß des AUTOSAR-Standards übernehmen verschiedene Basis-Software-Module des Steuergerätes diese Aufgabe. Welche Module zum Einsatz kommen, hängt von der benötigten Gateway-Funktionalität ab:

### PDU-Gateway

Das PDU-Gateway ist ein Teil des PDU-Routers (**Bild 3**). Dieser leitet komplette Datenpakete, die so genannten Protocol Data Units (PDUs), von einem Netzwerk auf ein anderes weiter. Dieses Prinzip setzt voraus, dass die PDUs sowohl auf dem Quell- als auch auf dem Zielnetzwerk gleich definiert sind, d.h. in Länge und Inhalt übereinstimmen. Damit lassen sich auch Daten zwischen unterschiedlichen Bussystemen wie CAN, LIN oder FlexRay austauschen. Allerdings muss man beachten, dass gemäß der AUTOSAR-Spezifikation die PDU bei Empfang direkt weitergeleitet wird. Dadurch ist der PDU-Router nicht in der Lage, eine Umsetzung von Sendeart oder Zykluszeit durchzuführen. In manchen Fällen wird diese Umsetzung allerdings benötigt. Ein Beispiel hierfür ist ein FlexRay-CAN-Gateway, das eine PDU von einem FlexRay-Cluster als CAN-Botschaft auf ein CAN-Netzwerk weiterleitet. Hier muss das Gateway-Steuergerät beispielsweise den Mindest-Sendeabstand der CAN-Botschaft einhalten. In solchen Fällen werden die PDUs deshalb direkt an die COM-Schicht geleitet, die diese Aufgabe übernimmt.

### TP-Gateway

Eine weitere Aufgabe des PDU-Routers ist das Weiterleiten von Transportprotokoll-Daten. Dies spielt dann eine Rolle, wenn etwa umfangreiche Diagnosedaten eines Steuergerätes automatisch von einem Netzwerk auf ein anderes übertragen werden müssen. Hierbei werden die Daten über das Transportprotokoll (TP) empfangen und wieder gesendet. Die Übertragung erfolgt also erst oberhalb des TP (Schicht 4 im ISO/OSI-Schichtenmodell) und erlaubt eine Umsetzung auf unterschiedliche Adressierungsverfahren und diverse Bussysteme. Um die Verzögerung und den notwendigen RAM-Bedarf im Gateway möglichst klein zu halten, unterstützt das TP-Gateway das so genannte „on the fly routing“. Das Gateway wartet nicht erst den Empfang der kompletten TP-Daten ab, sondern beginnt bereits zu einem früheren Zeitpunkt mit dem Weitersenden. Es empfängt und sendet also gleichzeitig.

### Signal Gateway

Oftmals werden nur einzelne Signale auf dem anderen Netzwerk benötigt. In diesem Fall überträgt das Gateway nicht die gesamte PDU, sondern nur einzelne Signale auf den anderen Bus. Dafür zerlegt es eine empfangene PDU erst in die einzelnen Signale, um diese dann in eine oder mehrere Sende-PDUs zu übernehmen. Neben der geänderten Signalzusammensetzung und Signalpositionierung innerhalb einer PDU lassen sich dadurch auch die Sendeart und die Zykluszeit verändern. Dieses Verfahren erfolgt auch dann, wenn eine PDU sowohl weitergeleitete Signale als auch direkt im Gateway-Steuergerät erzeugte Signale enthalten soll.

## Technische Aspekte bei der Post-Build-Konfiguration

Datenstrukturen für die post-build-konfigurierbaren Daten sind grundsätzlich auf zwei unterschiedlichen Arten aufgebaut (**Bild 4**): In der nicht fragmentierten Variante 1 sind die Datenstrukturen direkt hintereinander im Speicher angeordnet. Über eine Indirektionstabelle an einer statischen Position erfolgt der Zugriff auf die einzelnen Datenstrukturen, die auf Grund der Post-Build-Konfigurierbarkeit eine variable Größe haben können. Der verbleibende Speicher ist ein zusammenhängender Block, der für weitere Zwecke zur Verfügung steht. Es besteht allerdings auf der Implementierungsebene eine starke Abhängigkeit

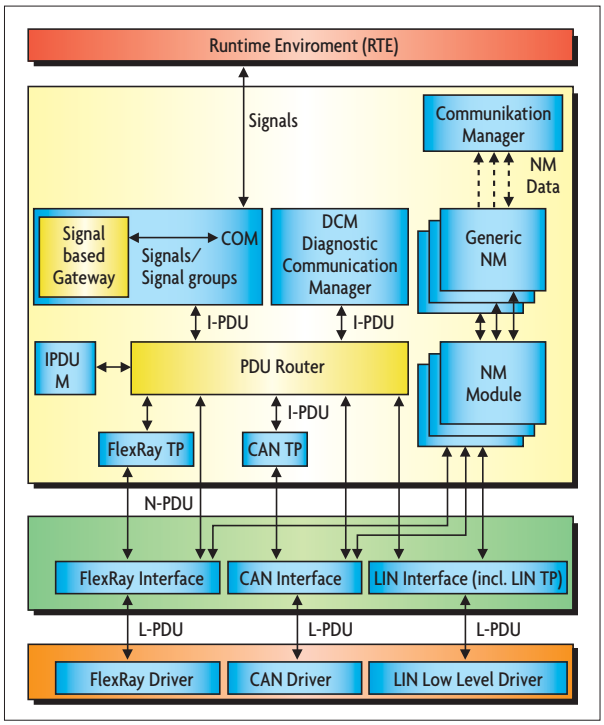


**Bild 3. Gateway-Funktionalität wird in der AUTOSAR-Software-Architektur durch den PDU-Router oder das COM-Modul realisiert.**

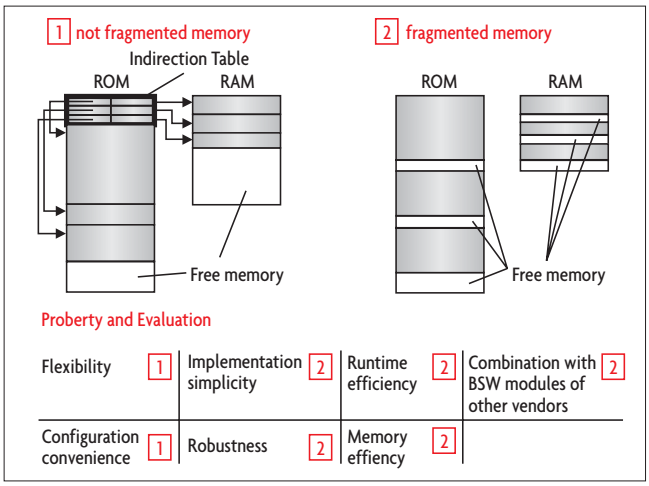


**Bild 4. Ein Vergleich der Datenstrukturen für eine Post-Build-Konfiguration zeigt Vorteile für die Nutzung der Variante ohne Indirektionstabelle.**

der Basis-Software-Module untereinander. Stammt die Basis-Software von unterschiedlichen Software-Zulieferern, erzeugt diese Variante einen hohen Abstimmungsbedarf und ist deshalb nicht sinnvoll einsetzbar.

In der fragmentierten Variante 2 sind die Datenstrukturen stets an einer statischen Position im Speicher platziert. Hierbei wird zum Entwurfszeitpunkt eine Annahme über den größtmöglichen Speicherverbrauch der einzelnen Datenstrukturen getroffen. Im konkreten Fall bleibt damit zwischen den Datenstrukturen üblicherweise freier Speicher übrig. Bezüglich des Laufzeitverhaltens ist die Variante 2 besser geeignet, da keine Indirektion beim Datenzugriff erforderlich ist. Trotz der Fragmentierung bietet diese Variante auch bezüglich der Speicherausnutzung Vorteile, da die Indirektionstabelle entfällt.

Die Verwendung von Post-Build-Datenstrukturen hat eine erhebliche Auswirkung auf den Entwurf der Basis-Software-Module. Im Fall der Pre-Compile-Konfiguration entfällt z.B. eine Trennung zwischen Code und Daten. Dadurch lassen sich Konfigurationseinstellungen sehr einfach mit Makros oder Präprozessorschaltern realisieren und mit Hilfe von Code-Generatoren optimierte C-Funktionen generieren. Das Prinzip der Post-Build-Konfiguration hingegen verlangt für Post-Build-Parameter eine strikte Trennung von Code und Daten. Ein Generator steht nur zur Erzeugung von Konstanten-Tabellen zur Verfügung. Die C-Funktionen sind statisch. Die **Tabelle** zeigt anhand von Code-Beispielen, wie die unterschiedlichen Konfigurationskonzepte umgesetzt werden.

| Pre-Compile-Zeitpunkt | Post-build/link-time |
|---|---|
| #define FEATURE_ENABLED | static boolean FEATURE_ENABLED; |
| #if defined(FEATURE_ENABLED) | if (FEATURE_ENABLED == true ) |
| ... | { |
| #endif | ... |
| | } |
| #define VALUE 8 | const uint8 value = 8; |

**Code-Beispiele für unterschiedliche Konfigurationskonzepte**

Ein Gateway-Steuergerät verarbeitet große Mengen an Signalen oder Botschaften. Diese Informationen befinden sich in Form von Datenstrukturen im Speicher des Steuergerätes. Folglich nehmen bei Gateway-Steuergeräten die Kommunikationsschichten in der Software-Architektur einen maßgeblichen Teil der Speicher- und Laufzeit-Ressourcen in Anspruch. Auch bei Wahl der Variante 2 ergibt sich durch die post-build-fähige Auslegung des Steuergerätes typischerweise ein erhöhter Ressourcenbedarf.

## Eine Werkzeugkette für die Post-Build-Konfiguration

Neben den Aspekten der Speicher- und Laufzeit-Ressourcen im Steuergerät erfordert das Post-Build-Prinzip auch neue Prozesse, um Konfigurationsparameter zwischen den beteiligten Entwicklungspartnern abzustimmen und auszutauschen. Eine wesentliche Hilfe stellt hierbei eine gut funktionierende Werkzeugkette dar. **Bild 5** zeigt die Werkzeugkette der Vector Informatik, die auch für die Post-Build-Konfiguration von Gateway-Steuergeräten eingesetzt werden kann.

Zu Beginn eines Entwicklungsprojektes setzt der Steuergeräte-Lieferant auf Basis des Parametersatzes „Pre-Config1“ das Projekt auf. Dieser Parametersatz wird passend zur Basis-Software-Lieferung von Vector zur Verfügung gestellt und enthält die Pre-Compile-Parameter, die keinen Einfluss auf die Post-Build-Konfiguration haben. Beispiele hierfür sind generelle Einstellungen wie die Verwendung des Development Error Tracer (DET).

In Abstimmung mit dem Steuergeräte-Hersteller liefert der Fahrzeughersteller den Parametersatz „Pre-Config2“ und eine initiale Netzwerkbeschreibung als Datenbasis. Der Pre-Config2-Parametersatz enthält diejenigen Pre-Compile- und Link-Time-Parameter, die Einfluss auf den Post-Build-Prozess haben und vorab festgelegt werden müssen. Beispiele hierfür sind die Adressen der Post-Build-Daten im Steuergerät, die Compiler-Optionen und die maximale Größe des Speichers (Flash und RAM). Die initiale Netzwerkbeschreibung, die der Fahrzeughersteller beispielsweise mit dem Werkzeug DaVinci Network Designer erstellen kann, enthält alle für das Steuergerät relevanten Signale. Im Fall eines Gateway-Steuergerätes sind dort auch die Routing-Beziehungen zwischen den Netzwerken beschrieben.

Der Steuergeräte-Hersteller nutzt das Werkzeug GENy, um mittels dieser Eingangsinformation die Basis-Software zu konfigurieren und zu generieren. Die Routing-Information wird dabei in Form von generierten Datenstrukturen (Tabellen) für die einzelnen Basis-Software-Module aufgearbeitet. Auf dieser Grundlage entsteht ein funktionsfähiges Steuergerät gemäß der initialen Netzwerkbeschreibung.

Während des eigentlichen Post-Build-Prozesses müssen diese Tabellen neu generiert und im Steuergerät ausgetauscht werden. Basis hierfür ist eine geänderte Netzwerkbeschreibung des Fahrzeugherstellers. Liegen die aktualisierten Tabellen nun binär vor – und zwar in genau der gleichen Form, wie sie der Compiler angelegt hätte –, kann auf einen erneuten Compilier- und Linklauf verzichtet werden. Die dafür benötigten Informationen bezüglich des Verhaltens des Compilers sind in GENy hinterlegt. Diese Binärdatei wird nun in ein hexadezimales Standard-Format konvertiert und über den Flash-Bootloader CANfbl in das Steuergerät geladen. Wenn dem Fahrzeughersteller die Pre-Config-Informationen bekannt sind, kann er den Post-Build-Prozess auch direkt durchführen.

Der Post-Build-Prozess bietet bei Änderungen in der Kommunikations-

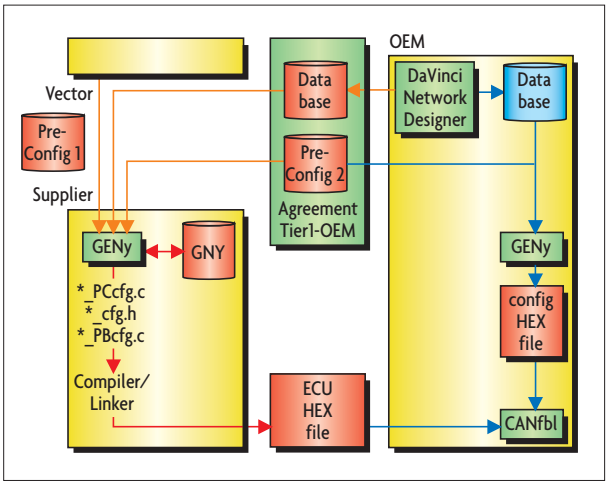


**Bild 5. Mit der Vector-Werkzeugkette lassen sich Steuergeräte auch zum Post-Build-Zeitpunkt konfigurieren.**

beschreibung die notwendige Flexibilität. Die Konfiguration ist sogar in späten Entwicklungsphasen während der Steuergeräte-Integration oder im Feld möglich. Insbesondere für Gateway-Steuergeräte ist dieser Ansatz nützlich, da man sie an geänderte Netzwerkgegebenheiten anpassen kann, ohne die Applikation komplett umzustellen. Einen höheren Ressourcenbedarf muss man allerdings in Kauf nehmen. Während der Entwicklungsphase ist ein Gateway-Steuergerät auf jeden Fall ein interessanter Kandidat für den Post-Build-Prozess. *fr*

### Literatur

[1] **Offizielle AUTOSAR-Website:** www.autosar.org
[2] **Spezifikation des AUTOSAR-OS, Version 2.0:** www.autosar.org/download/AUTOSAR_SWS_OS.pdf
[3] ***Patzer, A.:*** **Richtiges „Flashen“.** www.elektroniknet.de/home/automotive/technik-know-how/uebersicht/l/test-entwicklungstools/richtiges-flashen-fuer-jede-aufgabenstellung/
[4] **Design- und Entwicklungswerkzeugensowie Software-Komponenten für AUTOSAR:** www.autosar-solutions.de

**Dipl.-Ing. Hartmut Hörner**
studierte von 1987 bis 1992 Elektrotechnik an der Universität Stuttgart. Anschließend arbeitete er als Software-Entwickler für ATM Test Systeme. 1998 kam er zu Vector und ist hier als Teamleiter für die Entwicklung von Embedded Software-Komponenten zuständig. Er ist für Vector in verschiedenen Arbeitskreisen (OSEK, ISO, AUTOSAR) aktiv.
hartmut.hoerner@vector-informatik.de