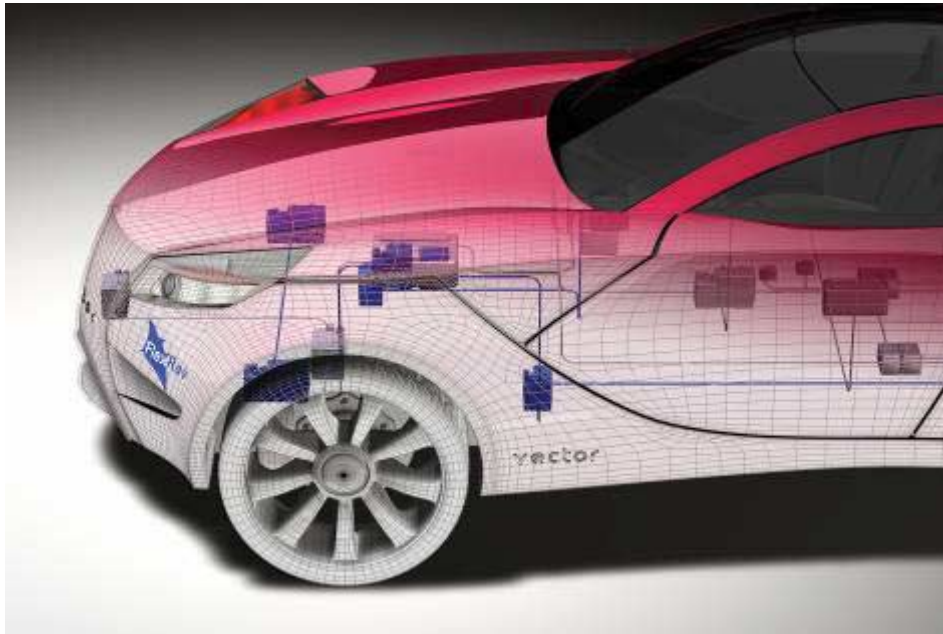


## Embedded Software for FlexRay Systems

Special aspects and benefits of implementing modularized software

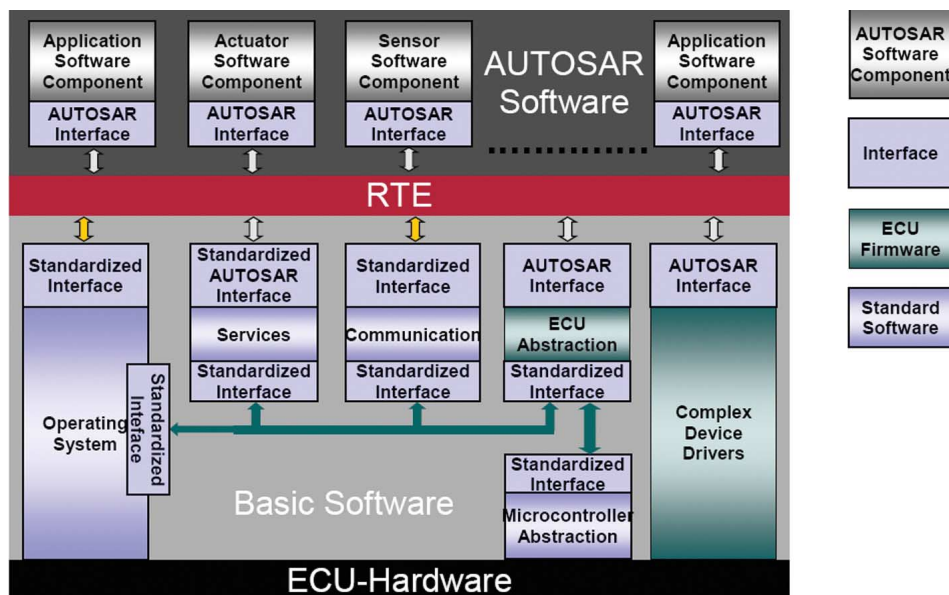


Standardized software components will help in mastering the growing complexity of the interplay of all software components in an ECU. The ways in which today's ECU software should be developed for FlexRay were presented at the Vector FlexRay Symposium in March of this year.

The FlexRay bus protocol is an in-vehicle network on its way to the starting line providing large transmission bandwidth for fast control systems. FlexRay enables 20 times greater bandwidth than CAN and reduces complexity by using fewer gateways. Because of its time-triggered control, it is especially well suited as a communication system for distributed, fault-tolerant systems and safety-relevant applications. There is also the hope that the growing complexity of vehicle electronics can be kept in check by standardization of the software system architecture with AUTOSAR.

## What does FlexRay ECU software need to look like?

To fully exploit the advantages of FlexRay-based communication, it makes sense to fundamentally develop the associated basic software according to the AUTOSAR specification. AUTOSAR specifies a new development methodology, software architecture and basic software. OEMs may gradually introduce it step-by-step on new vehicle models. The standard-conformant ECU-specific software is modularly structured (Figure 1). This enables partitioning of the software components above the RTE and the basic software below the RTE. The basic software has a modular internal structure and is specified by clearly defined interfaces, so that software from any source can be used in integration. In addition, the standard defines which exchange formats can be used and how the interfaces between individual modules need to operate.



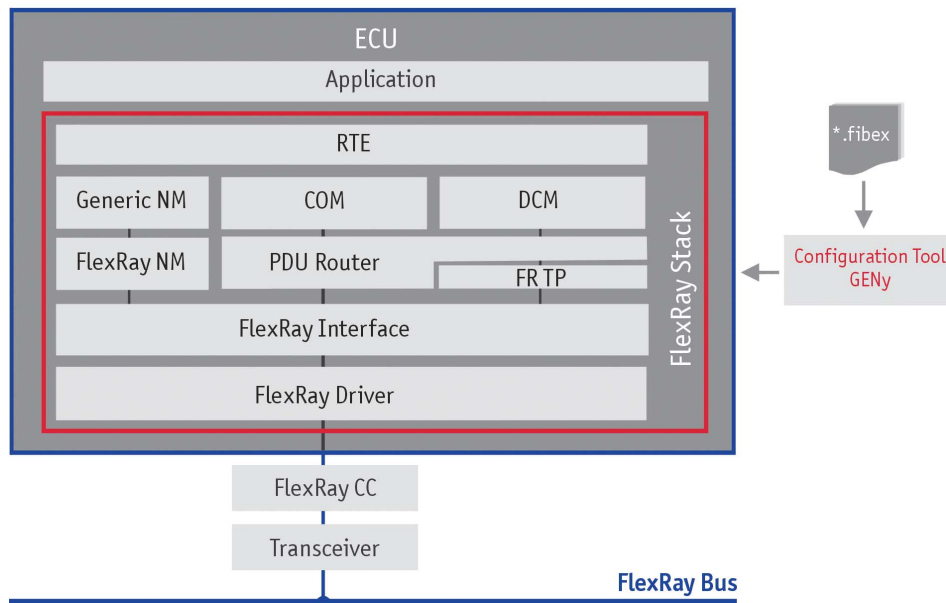
[Figure 1: AUTOSAR layer model of ECU software with modular structure.]

This modularity makes it easier to scale software features to the specific requirements of a vehicle variant or

generation, e.g. ECUs without network management. Development effort for the basic software can be reduced for both OEMs and ECU suppliers because the individual software modules can be delivered fully preconfigured by the software supplier. This lets developers place greater focus on innovations and the actual development of functions than was previously possible in development.

### **Embedded Architecture for FlexRay**

The schematic layout of FlexRay basic software from Vector is depicted in Figure 2. FlexRay-specific components such as the interface, driver, network management (NM), and transport protocol (TP) are all contained in the FlexRay stack. The driver abstracts the hardware, making it possible to service different communication controllers (CC). The driver initializes the controller, sends and receives frames and detects controller errors. The interface communicates with the layers lying above it, processes PDUs (Protocol Data Units) into frames and works in the reverse direction too. Moreover, it issues send and receive acknowledgments to the affected layers. Network management handles the coordination of all ECUs in the cluster with regard to communication needs for the bus system. If all of the bus nodes no longer have communication needs, the synchronous transition to the Bus Sleep ECU mode is initiated.



[Figure 2: Schematic layout of FlexRay basic software from Vector.]

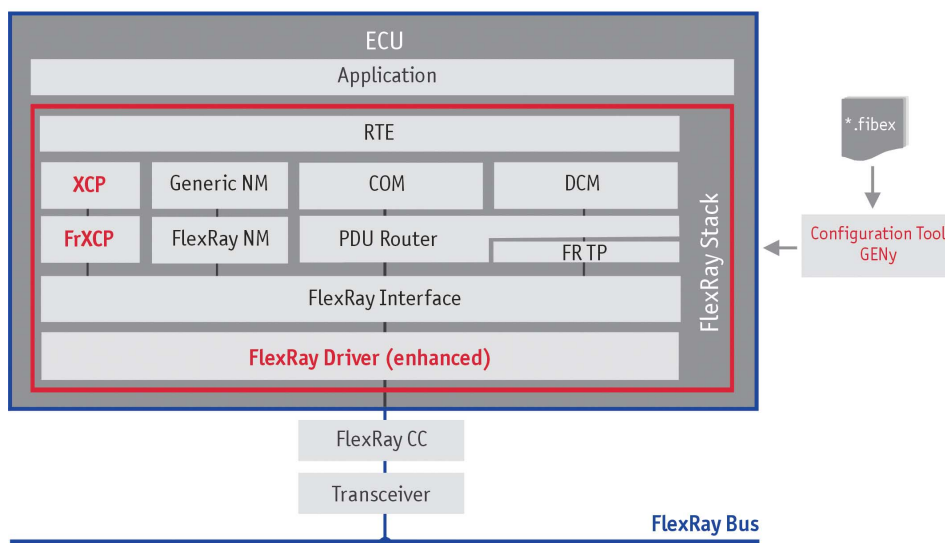
The FlexRay transport protocol is also placed on the FlexRay interface. It handles the task of taking large data packets that cannot be sent in one PDU, breaking them down into segments, and reassembling them on the receiving side.

Just how modular adaptation works in practice is demonstrated by the example of two FlexRay drivers for a FlexRay controller from Freescale and from NEC. The driver is optimally adapted to the specific hardware used and utilizes its existing features while still providing the layer above it with an unchanged "view" and mode of behavior. In the case of the 16-bit S12X controller from Freescale, the FlexRay driver must manage the buffer-RAM, since in this case the necessary memory space must be swapped out to the system-RAM. The 32-bit NEC V850 controller already contains a large RAM for the buffers in the FlexRay controller. This is where the driver performs its efficient partitioning and utilization.

## ECU calibration with XCP on FlexRay

It is even easy to integrate components developed at a later time in the architecture, e.g. components that became necessary because of new protocols or extended standards. These interfaces must also conform to the AUTOSAR standard. For example, XCP functionality might be added to the previously described FlexRay stack in order to enable measurement and calibration of internal signals of the FlexRay ECUs.

XCP is a universal communication protocol for optimizing the system parameters of an ECU. Due to the separation of protocol layer and transport layer, XCP can be operated in various types of communication networks (XCP on CAN, FlexRay, Ethernet, USB, RS232 or SPI/SCI). The clear separation of layers is also reflected in the integration in the FlexRay stack. The universal protocol layer XCP lies above the FlexRay-specific transport layer (FrXCP), which in turn enables signal exchange with the FlexRay interface (Figure 3).



[Figure 3: Integration of XCP in the FlexRay stack with clear separation of protocol layer (XCP) and transport layer (FrXCP).]

Due to dynamic bandwidth allocation, the driver must handle the additional task of buffer configuration during measurement or calibration. Therefore, this module is replaced by an extended version of the standard AUTOSAR driver.

XCP is an address-oriented protocol. Communication takes place between a controller component and a similarly structured software component in the XCP Master. Generally, the XCP Master is a measurement and calibration tool (such as CANape from Vector) that accesses measurement data in an address-oriented way, which are then acquired task-synchronously (event driven) in the ECUs.

### **Dynamic bandwidth management**

A connection is established via an initial communication channel that is defined in the ECU description file (A2L). By transport layer commands, the XCP Master controls allocation of available slots in the dynamic section of a cycle and thereby enables channel extension for transmission of measurement and calibration data. This "load distribution" is performed dynamically at runtime for optimal bandwidth utilization. Since multiple ECUs communicate over the same bus, a slot cannot be assigned exclusively; rather so-called slot multiplexing makes it available to multiple ECUs. This makes sense if less bandwidth is needed for a measurement, e.g. if an ECU does not need to send a message each cycle. Each message gets a unique address for this purpose (LPDU-Id, Data Link Layer Protocol Data Unit Identifier), which describes the slot,

cycle and channel. This makes it possible to fill the same slot with data from a different ECU in each send cycle.

The measurement data are provided with a timestamp making it possible to query measurement values at shorter intervals than the cycle time. For example, a measurement may occur every 2.5 ms, even if a system has a 5 ms cycle. It is only necessary to ensure that the bandwidth is sufficient to transmit the data within a required time.

The ECU must reserve and dynamically configure transmission and reception buffers by allocating RAM for this communication. The RAM may be located in the controller or it may be necessary to use external memory managed by the FlexRay driver.

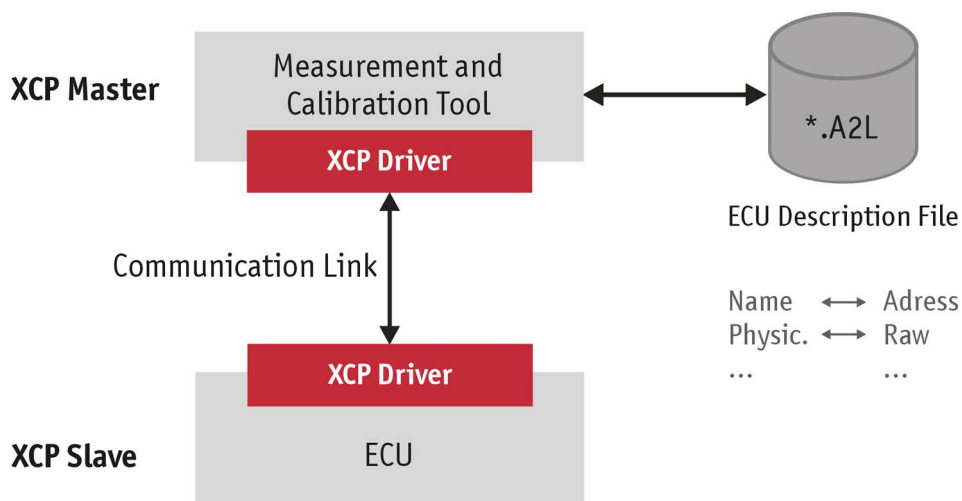
Just how many slots are available for XCP and how slots should be distributed must be defined early on - namely during system definition. This is done in the FIBEX file by also defining the slots reserved for XCP. Various scenarios are possible here:

- Each ECU occupies its own slot
- Multiple ECUs utilize the same slot  
In this case each ECU has its own address (NAX, Node Address for XCP). XCPtransport layer commands are used to activate or deactivate the buffer
- Buffers are configurable, which corresponds to dynamic allocation

Regardless of the transport protocol used, the user interface of a tool (XCP Master) should always represent the actual measurement and calibration task adequately. Applications engineers can then optimally tune ECU

parameters independent of the bus system used for communication.

The calibration engineer works with symbolic names, which allows the parameters and measurement variables to be found in the ECU without needing to know the encoded addresses of specific variables. The calibration tool uses the ECU description file to come up with the link between the name and the physical address (Figure 4).



[Figure 4: XCP is implemented as a master/slave structure. The XCP slave is located in the ECU; the XCP Master is located in the measurement and calibration tool.]

## Universal support in all development phases

The FlexRay developer gets universal support from the Vector company: From system description to implementation of the base software and calibration of the ECUs. This includes tools for design, development, simulation, analysis, testing of ECUs, and distributed networks and their bus interfaces. The first measurement, calibration and diagnostic tool to support XCP on FlexRay is CANape. With ready-to-use software stacks for ECUs and XCP-on-FlexRay support on the part of the calibration tool, the

user gets components that are perfectly coordinated with one another, are set up to be universal with AUTOSAR conformance, and can thereby be flexibly implemented.

---

Revised: 7/2007

Word count: 1,418

Character count: 9,196

### Figures:

Figure 1: AUTOSAR layer model of ECU software with modular structure.

Figure 2: Schematic layout of FlexRay basic software from Vector.

Figure 3: Integration of XCP in the FlexRay stack with clear separation of protocol layer (XCP) and transport layer (FrXCP).

Figure 4: XCP is implemented as a master/slave structure. The XCP slave is located in the ECU; the XCP Master is located in the measurement and calibration tool.

Figure 1: AUTOSAR GbR

Figures 2-4: Vector

### Authors:



Dirk Großmann (Graduate Engineer) studied electrical engineering at the University of Stuttgart. Since 2003 he has been employed as team leader responsible for the development of FlexRay embedded software components at Vector.

Tel. +49-711/80670-223, Fax +49-711/80670-111,

E-mail: [dirk.grossmann@vector-informatik.de](mailto:dirk.grossmann@vector-informatik.de)



Oliver Kitt (Graduate Engineer) studied physics at the University of Stuttgart. As team leader he is responsible for the development of measurement and calibration protocols in CANape.  
Tel. +49-711/80670-327, Fax +49-711/80670-111,  
E-mail: [oliver.kitt@vector-informatik.de](mailto:oliver.kitt@vector-informatik.de)

Vector Informatik GmbH  
Ingersheimer Str. 24  
70499 Stuttgart  
Germany  
[www.vector-informatik.com](http://www.vector-informatik.com)

Editorial contact person: Holger Heit  
Tel. +49-711/80670-567, Fax +49-711/80670-555,  
E-mail: [holger.heit@vector-informatik.de](mailto:holger.heit@vector-informatik.de)