

FLEXRAY TOOLS
 SUCCESSFULLY SUPPORT
 DEVELOPMENTS WITH PDU-
 BASED COMMUNICATIONS

AUTOSAR PDUs Conquer FlexRay

Audi is using FlexRay in their newest vehicles. The developed Flex-Ray network communication uses PDUs (Protocol Data Units) that are fully AUTOSAR compatible. PDUs are logical data containers for exchanging signals between applications and allowing greater decoupling from the underlying communication system. Audi benefited immediately from Vector's CANoe as a bus analysis and simulation tool which can natively handle PDUs.

With the release of the frame-oriented FIBEX 2.x database format, a new description semantic was needed to define PDU communications between network nodes. To overcome this gap, Audi successfully developed the FIBEX+ description semantic. Vector was able to immediately support FIBEX+ in its tools. Profiting from their experiences with FIBEX+, Audi introduced PDUs into the new FIBEX 3.0 standard from ASAM (Association for Standardisation of Automation and Measuring Systems [1]). Continuous feedback from Audi enabled Vector engineers to integrate important PDU features during the early phases of

tool development. Service Packs were delivered regularly to Audi, thus, allowing early testing of ECUs with PDU communication stacks. Audi delivered their latest FIBEX+ database versions to Vector in order to ensure CANoe's continuous compatibility. Close collaboration between Audi and Vector accelerated the tool development and led to a professional analysis and development platform for FIBEX+ and the new FIBEX 3.0 based FlexRay networks. This article illustrates the impact PDUs have on the internal structure and features of a FlexRay development tool CANoe.FlexRay and how Audi engineers benefit from appropriate tool support.

PDU Layer for Network Analysis

PDUs are managed by tools for analysis, simulation, and test as high-level communication data containers (e.g. messages) containing signals. A FlexRay frame can contain several PDUs. Since the layout of a frame can also change from cycle to cycle, the same PDU can be mapped to multiple frames. PDUs are uniquely identifiable by their position in a FlexRay frame in a specific slot during a specific cycle. Vector identifies PDUs in CANoe via the PDU Layer (Figure 1). The PDU Layer introduces PDU objects and is located between the bus and the user interfaces, respectively. The layer is enabled or disabled according to the assignment of an appropriate database (FIBEX+ or FIBEX 3.0) to CANoe. If the layer is enabled, then the complete symbolic database interpretation (PDU names, signals, timings, etc.) of the network communication is performed at PDU level.

The PDU's main property, which is defined by the Update Bit, is decoupled from the frame's occurrence on the network. Thus, frames on the network may contain updated and non-updated PDUs at the same time. Update Bit values can be visualized as pre-defined signals or can be evaluated (e.g. in the graphics window, see Figure 2). As a default for simple analysis or simulation received PDUs, which have not been updated, are ignored. For a detailed analysis, however, non-updated PDUs can optionally be displayed and passed on to the simulated nodes. In addition, FlexRay frames including their payload can be displayed and received as so-called Raw Frames. Such PDU-based analysis features were heavily used by Audi during their integration tests.

PDU Layer for Network Simulation

Although the FlexRay protocol defines frames to be transmitted cyclically (even without any update), native PDUs do not have this property. If a PDU is not updated, the receiver will normally not recognize the PDU. In order to trigger the receiver cyclically, PDUs must be updated periodically. If the automatic transmission of PDUs with the Update Bit set (i.e. without any explicit data update) is required, then the network designer can define these PDUs to have cyclic timings. For this reason an Interaction Layer (see Figure 3) on top of the PDU Layer was developed to handle those constraints. As an extension to the FlexRay protocol, PDUs may also be sent cyclically with (nearly) arbitrary periods with set Update Bit (without being updated).

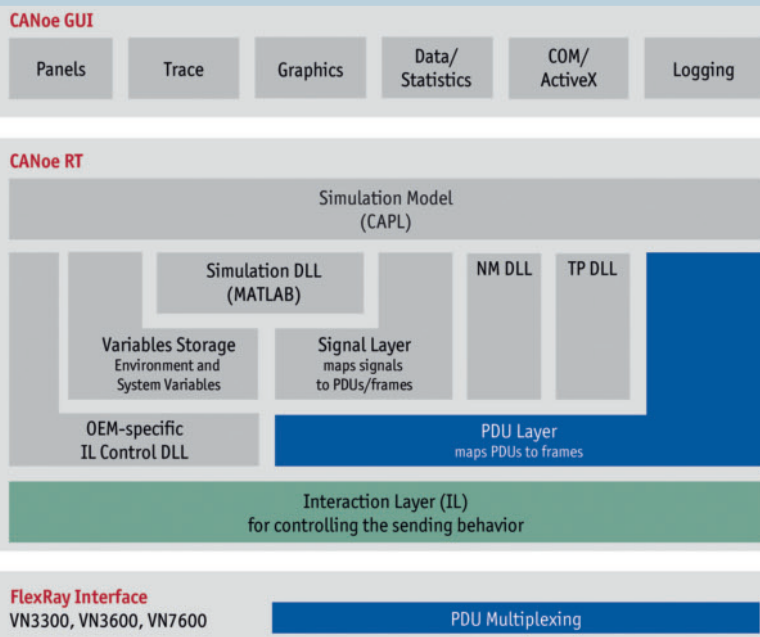


Figure 1: CANoe's Architecture with Abstraction Layers for Signal and PDU Handling and Sending Behaviour (IL).

© automotive

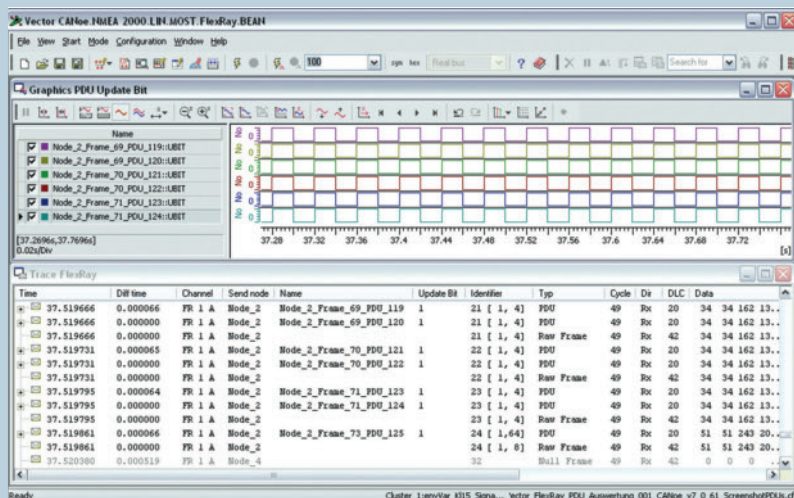


Figure 2: CANoe's visualization of PDU's Update Bits in Graphics and Trace.

© automotive

Message counters and check sums were defined by Audi as additional but optional validity attributes of a PDU. In fact, the Update Bits, message counters, and check sums of a PDU are set independently from the application in CANoe by the Interaction Layer in order to simplify the remaining bus simulation. Thus, the engineer can concentrate on setting appropriate signal values. A further use case is the insertion of explicit failures into the remaining bus simulation in order to test an ECU's reaction. Therefore every automatic feature in CANoe can be disabled and the Interaction Layer can be used for fault injection. A simulation of communications with an ECU depends on the occurrence of specific events (event-based simulation).

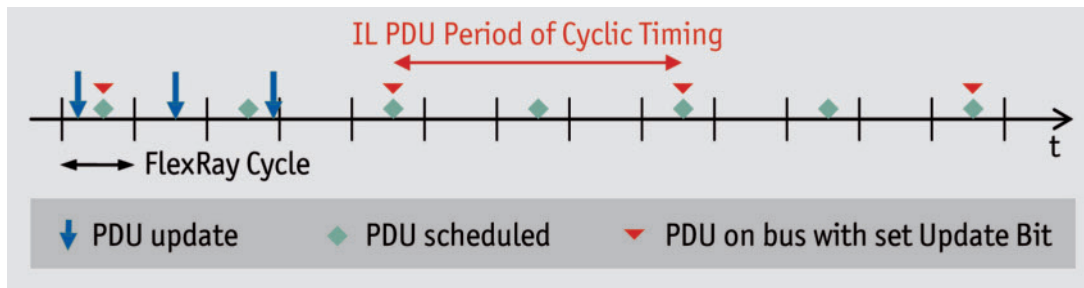


Figure 3: The Interaction Layer (IL) of CANoe controls the sending behaviour of PDUs with set Update Bit inclusively their extended validity attributes (message counter, user CRC).

© automotive

One of the most important events is the reception of messages or changes in signal values from the bus. As such, notification handlers for PDU reception and signal changes are triggered by the PDU Layer.

Performance Aspects

The additional (but mandatory) PDU Layer does create some overhead. When receiving FlexRay frames, PDUs have to be extracted from the frame and then forwarded to their notification handlers in the application. The same PDU can be contained in different frames. Thus, a sort of de-multiplexing of PDUs from frames is implemented by the PDU Layer. These procedures are highly optimized.

On transmitting PDUs, they must be stored in the appropriate frame. The PDU can be located in different frames according to the current (cycle) time or a different set of PDUs may be contained in the frame. This results in a highly time-dependent multiplexed mapping of PDUs to FlexRay frames. If this is not fast enough, then frame slot misses should be expected. For maximum performance, Vector has implemented those functions to run on the hardware for the FlexRay bus interfaces of the VN series.

Testing PDU-based Networks

Audi and its Tier1 suppliers also benefited from CANoe's AUTOSAR features. This includes a check for communication conformance in order to test the AUTOSAR communication stacks (especially the PDU Router) of the ECUs. Here it is of great importance to be able to compare the real bus entities (Raw Frames) with the symbolic interpretation (PDU abstraction level). This helped the Audi engineers to identify in early phases an incorrect PDU or Update Bit position in the raw FlexRay frames.

Tests can be split into two categories: First the application's transmission behaviour can be checked with respect to updated PDUs and secondly, a signal's integrity can be

verified according to the application. Audi's engineers have successfully detected incorrect PDU update timings in early development phases. These tests are fully supported in CANoe's Test Feature Set for PDUs. Additionally, for stimulus and response observations, PDUs can be sent interactively (PDU Panel) by implicitly manipulating signals (Input Panels), higher level protocols (transport, diagnostics), or by a remaining bus simulation (CAPL, MATLAB models, etc.).

Conclusion

PDU-based communications will not only be used in migration scenarios, e.g. when porting applications from CAN to FlexRay networks, but also for new FlexRay developments. Audi has strongly influenced the development of FIBEX 3.0 based on lessons learned with FIBEX+. Vector's experience with FIBEX+ allowed the quick support of the new FIBEX 3.0 standard with CANoe. As communications become more complex and extensive, appropriate modelling standards (i.e. FIBEX 3.0 and/or AUTOSAR), as well as their professional tool support, are essential requirements for an efficient FlexRay development.

Literature:

- [1] www.asam.org



Dipl.-Ing. (FH) Wolfgang Brandstätter is Hardware and Protocol Engineer for FlexRay; AUDI AG, 85045 Ingolstadt, Germany



Dr. rer. nat. Carsten Böke is Senior Software Development Engineer for the "Tools for Networks and Distributed Systems" product line; Vector Informatik GmbH, 70499 Stuttgart, Germany