

Efficient Access to the FlexRay Bus

High-performance FlexRay Hardware for Analysis and Simulation

PC interfaces for various standards are indispensable tools in all development phases of automotive electronics, wherever access is needed to what is happening on the bus. OEMs and suppliers are being confronted with especially complex challenges with the current introduction of the FlexRay bus in first production vehicles. Much more than on the CAN bus, high performance hardware is a prerequisite for experiencing reliable operation in all situations and fully exploiting the potential of software tools for simulation and analysis.



The various tasks of FlexRay development and associated tests require interfaces for both stationary PCs and mobile notebooks (Figure 1). In both cases, they need to satisfy special requirements for simulation, analysis, calibration and testing. For example, an ECU's standard controller recognizes when errors occur, but does not provide any information whatsoever on their causes. For a qualified analysis, the developer needs – besides the FlexRay messages and signals themselves – precise time stamps, comprehensive information and detailed itemization of all states on the bus and in the interfaces. Compared to previous automotive bus systems, technical requirements for FlexRay are significantly more stringent. Since



Figure 1:
Hardware interfaces must also be implementable in mobile FlexRay applications.

FlexRay's operation is not event-driven, but instead is time-triggered, it is necessary to synchronize all bus nodes. Send times are precisely specified by the cyclic and synchronous time-division multiplexing TDMA (Time Division Multiple Access) bus arbitration.

FlexRay Interfaces for all Requirements

A new generation of FlexRay interfaces delivers the right solutions for all situations occurring in practice. In particular, one focus in their development was a high level of assurance of future utility. For example, updates to the FlexRay standard due to FPGA (Field Programmable Gate Array) technology are fed-in by customers themselves with driver updates.

On the one hand, behavior of the FlexRay interfaces from Vector conforms to the standard; on the other, they are able to log all conceivable bus events due to their supplemental FPGA logic. They record 100% of the bus traffic of both FlexRay channels. The centerpiece – an Intel PXA270 microcontroller together with 8 MByte RAM – is supported by the Bosch E-Ray and Fujitsu MB88121B FlexRay communication controllers. Interchangeable as plug-in modules, electrically isolated bus transceivers lend flexibility to physical bus access with regard to future requirements.

Optimized for Analysis

Overall, the interfaces were optimized for the best possible interaction with the simulation and analysis tools CANoe and CANalyzer, and the measurement and calibration tool CANape (Figure 2). The interfaces not only recognize all bus activities and buffer them as necessary; rather they also pass all information to the host. Different than on controllers for ECUs, the controller host interface is de-

signed to record all data, null and erroneous frames as well as symbols, including the associated timestamps and pass them on to the software tools. This is the only way for developers to analyze, meaningfully interpret and thereby systematically troubleshoot sources of errors. If no FlexRay synchronization is established or no FIBEX database is available with TDMA parameters, an asynchronous bus analysis is possible, in which it is only possible to follow events and log them in reading operation. In this mode, it is also possible to observe the startup phase of a FlexRay network. The measurement and calibration tool CANape gives developers the ability to access internal ECU parameters via the standardized XCP-on-FlexRay protocol. In this case, the FlexRay hardware supports resynchronization of the FlexRay interface if bus traffic has been interrupted.

Maximum send Throughput for Simulations

The requirements demanded by ECU simulations on the PC, e.g. with CANoe, are significantly greater than those for analysis mode. Since multiple ECUs can be simulated simultaneously on a sufficiently fast computer, the interface must also be able to handle the higher data throughput while taking all timing requirements into account. Parallel simulation of ten or more ECUs is entirely realistic. It is noteworthy that it only takes one of the new FlexRay interfaces to

achieve this. This performance is enabled by the TX buffer that has been extended to 2 MByte that can store over 1000 independent send messages. Previous solutions typically only had an 8 KByte TX buffer.

The CANoe RT platform is especially well-suited to small to mid-size projects with real-time requirements, such as hardware-in-the-loop simulations. It isolates visualization and control functions from the real-time simulation. The simulation is executed troublefree on a separate computer under Windows XP Embedded, and this guarantees reliable updates of send time points. The only interface that comes into consideration for this computer, called a RT Box or RT Rack, is a fast PCI interface such as the VN3300 (Figure 3).

For minimal response times and deterministic timing behavior of the application, short latency times and minimal jitters are absolutely essential. Besides the time required for the actual computations of the application, it is necessary to add the times for transport processes through the different communication layers. To achieve low PC loading despite this situation, DMA (Direct Memory Access) was implemented in the FlexRay hardware. DMA enables high transmission rates and simultaneously relieves the main processor and gives it more time for computations. Latency times were

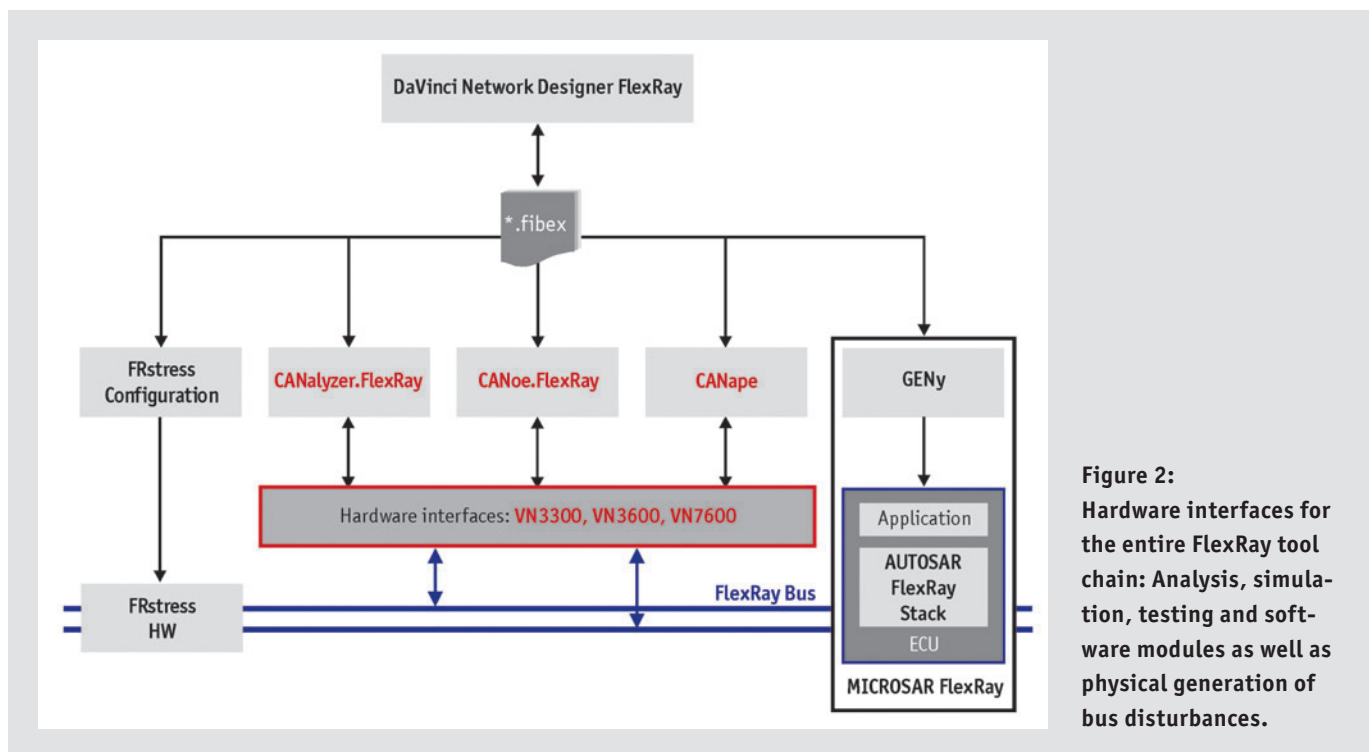


Figure 2: Hardware interfaces for the entire FlexRay tool chain: Analysis, simulation, testing and software modules as well as physical generation of bus disturbances.

optimized – as was already done on CAN interfaces from Vector. The shortest latency times are system-dependent and can be attained with PCI interfaces.

Intelligent auxiliary Functions for everyday Work of Developers

Experience and customer requirements on numerous FlexRay projects motivated developers at Vector to integrate a number of other important functions in the FlexRay interfaces: hardware support for PDUs, automatically incrementing message counters, simulation of inactive ECUs, group updates and autonomous bus start capability. To decouple the transport layer from the applications, in the latest FlexRay networks PDUs (Protocol Data Units) have been introduced, instead of working directly with the relevant bus-specific data containers. Several such logical PDUs may lie within a FlexRay frame. In this case, an additional piece of information is needed for each PDU, indicating whether or not the contents are valid for the current cycle. One and the same PDU may also be defined in multiple frames.

The PDU concept enhances flexibility and enables easy reuse of the application, but its disadvantage is that considerably more effort is required to generate and decode the FlexRay frames. Powerful FlexRay interfaces compensate for this disadvantage by handling

the multiplexing and demultiplexing of the PDUs into and out of the frames on the hardware side.

Hardware-based incrementing of a payload area serves to reliably simulate a sender's heartbeat. That is because, if it is not possible to guarantee this generation in a software-based simulation, the receiver might reject the signal or even switch itself off. The intelligent hardware prevents this by repeatedly sending the old signal values with counter incrementing, thereby reliably signaling that the sending device is still "alive".

Simulation of inactive ECUs enables later deletion and supplementing of frames to be sent, even if the user has not yet defined them at startup. The bus transceivers can be switched to inactive (Sleep mode), after which wakeup patterns are still detected, however. The bus transceivers can also actively execute a wakeup.

If data that belong together do not fit in a FlexRay slot, there is a risk that it might not be possible to consistently transmit the data in two frames of the same cycle. This is remedied by a "group update", wherein the interrelated frames are always sent together. To start up a FlexRay cluster, it is necessary to have at least two

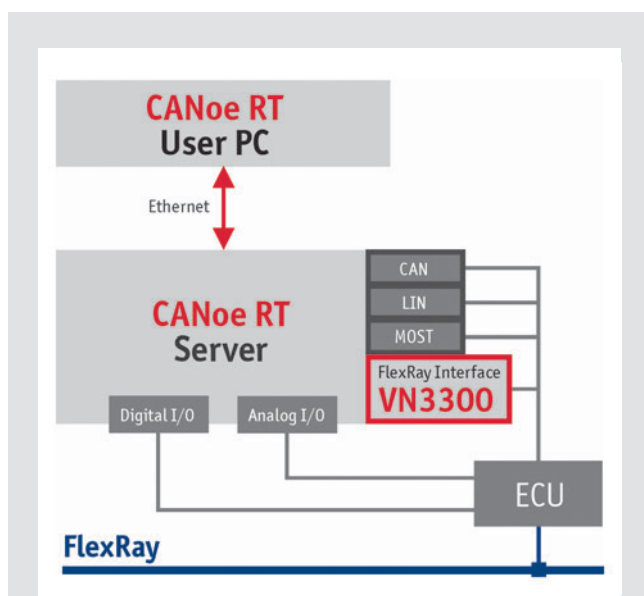


Figure 3: Stringent requirements are satisfied by the real-time capable and deterministic CANoe RT runtime platform.



Figure 4: VN interfaces from Vector fulfill the strict bus interface requirements imposed by the FlexRay bus. The USB variants VN3600/VN7600 are suitable for mobile applications, analyses and simple simulations. The PCI variant VN3300 supports complex simulations of multiple ECUs with real-time requirements.

ECUs that can execute a startup. Some ECUs are not startup-capable; they always integrate themselves in the communication after a successful external startup. If a network line only has these kinds of devices for the measurement or simulation, the bus system cannot be started up due to the lack of startup-capable nodes. Therefore, a second communication controller or startup controller has been integrated in all FlexRay interfaces.

Interfacing dedicated Applications with the Hardware

The new generation of FlexRay interfaces from Vector offers high-performance hardware solutions for the most significant PC platforms and interface types. These interfaces are specially tailored to the requirements in simulation, analysis, calibration and testing (Figure 4). The strengths of the USB variants VN3600 and VN7600 lie in the mobile area. They are ideally suited to analysis and simple simulations, while the VN3300 PCI card is intended for complex simulations involving multiple ECUs under real-time constraints. Currently, FlexRay buses are primarily used together with existing CAN networks. The VN7600 FlexRay/CAN-USB interface addresses this situation appropriately with two FlexRay channels and three CAN channels in one device. Developers of FlexRay/CAN applications benefit from simultaneous access to both bus systems in analyses and simulations with just one hardware module. The combined hardware solution for FlexRay and CAN especially simplifies precise synchronization of the different bus systems with highly precise time stamps and a common time base. In this regard, a significantly higher level of quality is achieved compared to multiple separate solutions, since latencies must always be expected when USB is used for interfacing.

A programming library of basic functions is supplied with the FlexRay hardware. This makes it possible for dedicated applications to access the Vector FlexRay hardware. The "Advanced FlexRay Driver Library" is available for extended functions. The developer uses this library to access extended functions of the interfaces, such as the second communication controller, extended TX buffer and automatic payload increment.

Summary

FlexRay places disproportionately greater demands on hardware and software than CAN or LIN networks, for example, due to its time-triggered transmission method and considerably higher transmission rates. Here, the timing behavior of the hardware has a decisive influence on the quality of software services that can be provided. Significant performance increases are realized by transferring software functions to the hardware.

In the area of FlexRay networking, Vector offers a universal tool chain, modular software modules as well as interface hardware and support for specific projects and a training program. The Stuttgart-based specialist's activities as a Premium Associate Member of the FlexRay Consortium ensure that advanced developments and the latest protocol specifications are always considered in the design of tools and hardware interfaces.



Dr. Carsten Böke

studied Information Technology at the University of Paderborn. From 1995 to 2004, he was employed as a scientific assistant at the Heinz Nixdorf Institute in the area of Parallel Systems Design. While there he was primarily involved with automatic configuration of embedded communication systems. Since 2004 he has been employed as a Senior Software Development Engineer at Vector Informatik GmbH, where he develops tools for bus analysis and bus simulation of FlexRay systems.



Alfred Kless

studied Electrical Engineering at the Technical College of Esslingen. From 1991 to 2004, he worked in the area of Test System Development at the company ALCATEL. Since 2004, he has held the position of Business Development Manager at Vector Informatik where he is responsible for the product lines "Network Interfaces" and "Measurement and Calibration".



Martin Goßner

studied Computer Engineering at the Technical College of Ulm. Since 1998, he has been employed in the "Network Interfaces" area at Vector Informatik GmbH. Since 2000, he has been team leader for driver and firmware development. As a product manager, he is responsible for Vector FlexRay interfaces.