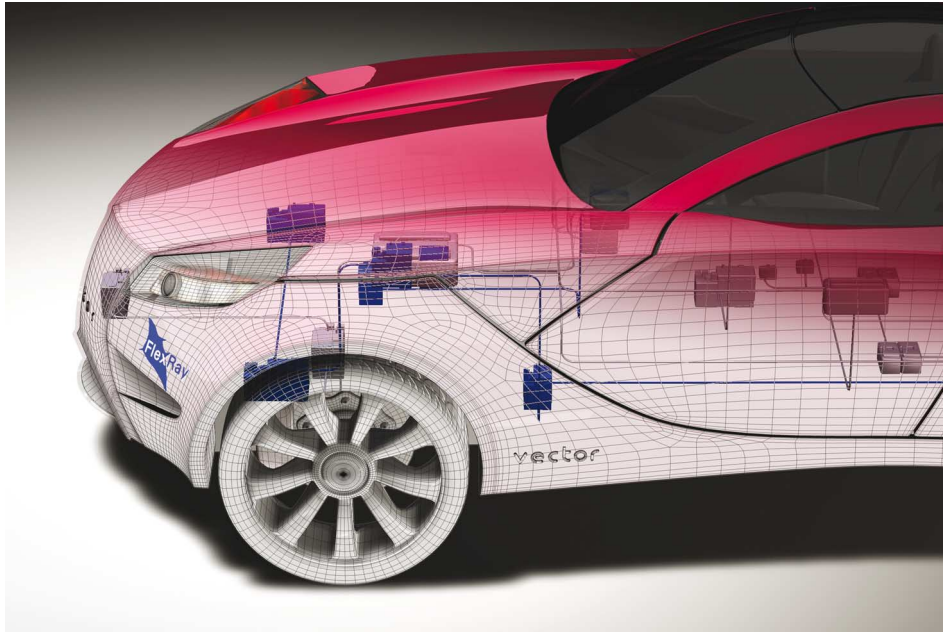


FlexRay is Driving

Partners demonstrate successful system development at the FlexRay Symposium



In March 2007, over 200 developers met in Stuttgart for the FlexRay Symposium sponsored by Vector Informatik. Automotive OEMs and suppliers outlined their successes to date, experiences in system integration and concepts for future implementations.

The CAN bus encountered its limits long ago. Modern automotive electronic architectures are continually requiring more extensive networking. Implementations of control algorithms that have become increasingly faster are only effective if additional transmission capacity is provided. On many car models, maximum bus loading is already reached at the start of production, leaving no bandwidth reserves. Doubling the number of bus systems does not in any way double the data rate. The additional

gateways that are necessary to network the systems not only lead to confusing complexity, but may produce unallowable delays in message transmission. And finally, the lack of determinism is a barrier to safety-critical applications.

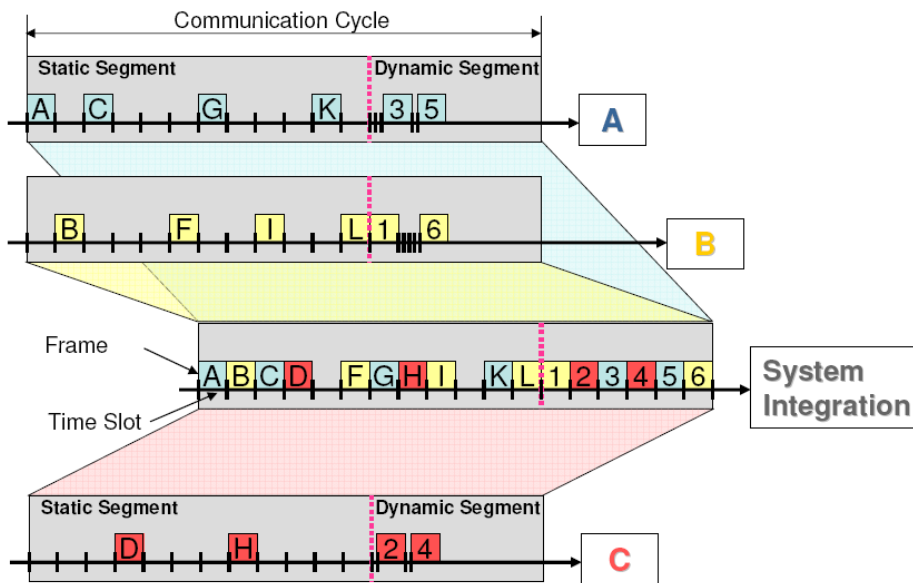
The certainty that CAN could not be expected to fulfill growing requirements for data transmission in the automobile over the mid-term led to the development of the FlexRay serial bus system [1]. At the end of last year, BMW presented the first production application of FlexRay. This was an ideal point in time for Vector Informatik to hold its FlexRay Symposium to offer an overview of the challenges and experience with the new protocol. Completion of the damper control system on the BMW X5 via FlexRay was a time-critical project in two respects, and it put the participating partners to the test. Not only did semiconductor products and software components need to be produced on time, but with such an ambitious project, the development process itself had to be quickly adapted to existing structures and it had to run properly. The support of suppliers is required here. "We could not develop a new development process at BMW just for FlexRay", says Dr. Anton Schedl, group leader for networking technologies at BMW AG. "Therefore a conscious decision was made to choose a relatively simple application, so that changes could be implemented quickly based on the experiences that were made and with short coordination and decision paths."

According to Dr. Schedl, the availability of semiconductors at the right time was the greatest challenge for the pilot project. Thanks to the aggressive commitments made by both the OEM and the semiconductor supplier, it was possible to achieve the objective on schedule.

All beginnings are difficult

Although starting up any new system is of course difficult, the various components were integrated relatively quickly. "Time-triggered communication is the ideal foundation for bringing together the components and software codes of different suppliers" states Florian Hartwich, who works in the area of networks for motor vehicles at Robert Bosch GmbH; he also assisted in work on the protocol in the FlexRay consortium and prior to that had participated in the development and standardization of CAN and TTCAN (Time Triggered CAN). Since each application sends messages at a precisely defined time point and knows when it needs to receive which messages, it is easier to add components to a distributed system at a later time.

The most important tasks need to be handled right at the start of development of a FlexRay system. All parameters that fundamentally describe the system - such as baudrate, cycle time, number of slots, slot length and distribution of messages to the static and dynamic segments - are defined at the beginning. Since fixed slots are assigned to the send tasks, it is already necessary to be clear in the project definition process about how the assignment of slots with messages will be structured, which applications might best be elevated to the dynamic event-oriented segment, and how many slots should be reserved for applications of subsequent car model series, Figure 1.



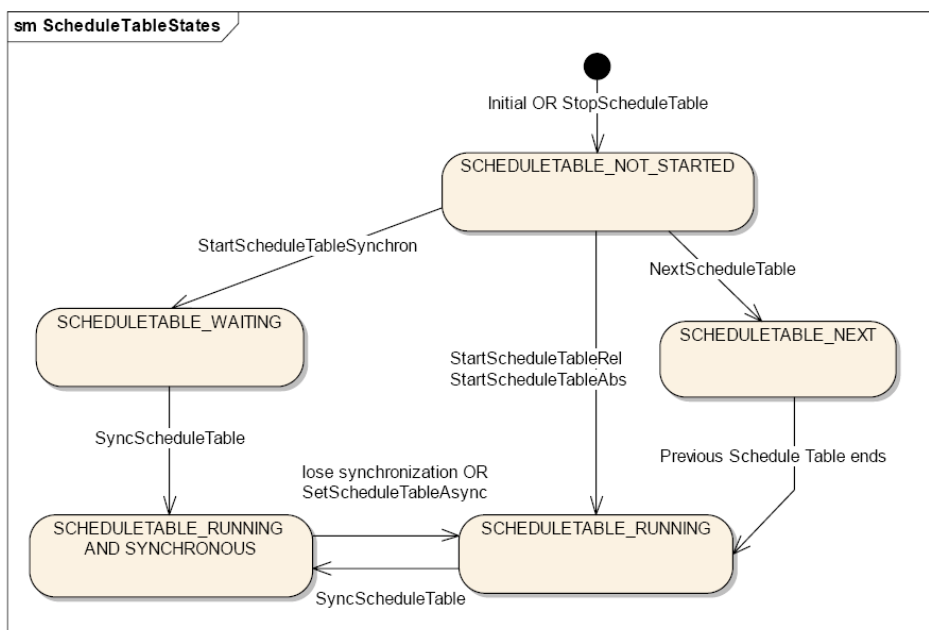
[Figure 1: Fixed allocation of transmission slots to individual components (A, B, C) simplifies their incorporation in system integration.]

It is especially important to maintain a general perspective in the case of distributed systems. At the beginning, network designers usually do not know any of the details of how the real applications will actually communicate at a later time nor the execution times they will have. ECU developers, on the other hand, like to concentrate exclusively on developing the application, without having to constantly be concerned about the time conditions of the overall FlexRay communication process. However, FlexRay data within a cycle must be consistent, and synchronism of the application with the time-triggered bus must be guaranteed at all times.

Hartwich therefore called attention to the problems that can cause data inconsistency. For example, it is essential to avoid updating send data during transmission; this could result in old data and new data within the same frame. BMW solved this problem by using so-called "signal windows" to ensure that a task only sends within this dedicated window.

Another benefit of this approach is that the application is decoupled from communication: If the communication schedule changes, this does not affect the application.

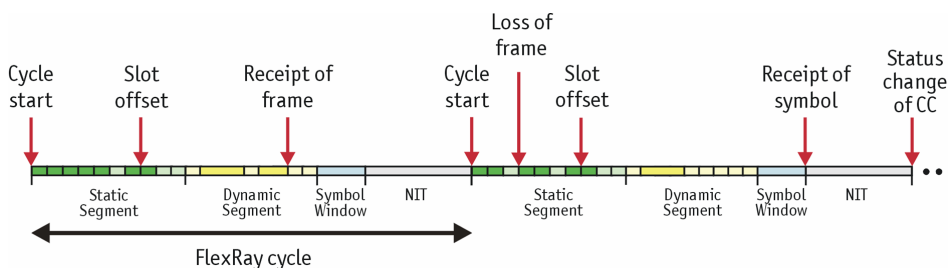
In a real-time capable system, synchronism of tasks is a key characteristic that must be given special attention. "The question of the proper synchronization of the tables is of crucial importance", explained Winfried Janz, team leader and product manager for the development of OSEK real-time operating systems at Vector. In his presentation on OSEKtime and AUTOSAR OS, he discussed the ways in which a schedule table can be synchronized to the global time according to specification, Figure 2. The choice between hard synchronization (a table jumps to a predefined execution point or is stopped briefly) and soft synchronization (time adjustments are made at each expiry point; but these points are distributed irregularly, resulting in somewhat irregular time adjustment behavior) is made based on whether or not the application tolerates jumps and pauses.



[Figure 2: The state diagram of a schedule table shows how synchronization is achieved. The schedule is started, but it need not be

absolutely synchronous right away (RUNNING). To run synchronously (AND SYNCHRONOUS) it can be put in a wait state (SCHEDULETABLE_WAITING), or it can be soft synchronized.]

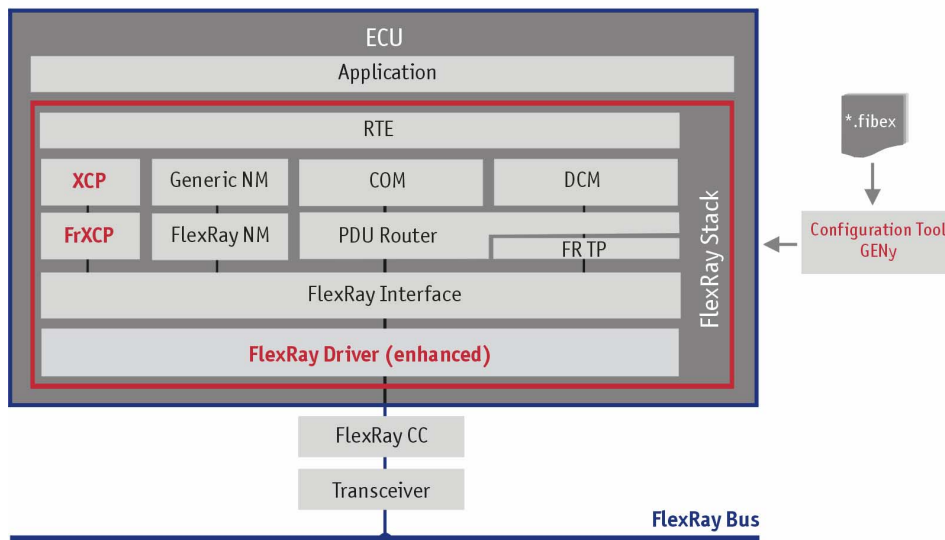
During the development phase, monitoring for synchronism and data consistency is the job of software tools. “We must be able to process a model synchronously, otherwise data would be lost,” said Dr. Carsten Böke as he explained a property of the Vector tool CANoe. Böke demonstrated the mechanisms the CANoe tool provides to ensure synchronism and to detect inconsistent data. CANoe’s fundamental architecture for model execution is based on a notification concept with so-called “notification handlers”. This encompasses model activation when messages are received, handling when timers expire, and detection of error states. In particular, this concept was extended for FlexRay to include synchronous notifications at specific time points in the FlexRay cycle, Figure 3. In addition, Böke presented an optimal platform with CANoe RT and special hardware support that is tailored to the stringent time requirements of FlexRay systems and is suitable for small and medium-size hardware-in-the-loop simulations.



[Figure 3: In CANoe, actions can be executed regularly in reference to the beginning of a cycle or after a specific slot has ended. Naturally, notifications are also possible when frames are received or are missing on the bus.]

FlexRay and AUTOSAR

"In order to be ready for the future, new software concepts must be designed according to AUTOSAR guidelines", said Dirk Großmann, who is responsible for the development of FlexRay basic software. The results and findings of the AUTOSAR consortium immediately flow into Vector's FlexRay development, since Vector Informatik is a member of the consortium, Figure 4. The FlexRay interface and FlexRay driver from Vector are already AUTOSAR-conformant, thus these components can be developed today, independent of their later specific application, and they are flexibly scalable to different vehicle and platform variants. The FlexRay driver abstracts the communication controller, and the FlexRay interface provides accesses to individual PDUs (Protocol Data Units) that are time-decoupled from the FlexRay schedule. Moreover, Vector offers AUTOSAR-conformant implementations for network management and the transport protocol. As a supplement to AUTOSAR, the XCP protocol can be integrated in the FlexRay stack. Großmann also mentioned the possibility of flash programming via FlexRay. "One scenario is to exchange the protocol completely and to use a separate schedule for flashing."

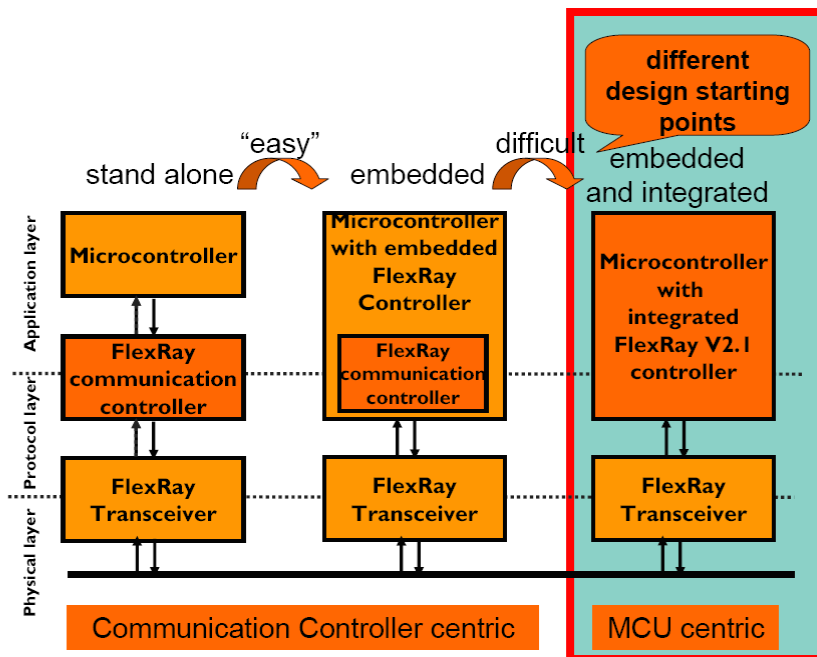


[Figure 4: Layout of the FlexRay software according to AUTOSAR enables simple adaptation to different requirements. The figure shows a FlexRay stack, with a driver that is extended relative to AUTOSAR and is supplemented by a XCP transport layer and protocol modules.]

Oliver Kitt, in his presentation, went into more detail on the topic of calibrating ECUs by XCP (a calibration protocol standardized by ASAM). At Vector he is responsible for integration of hardware interfaces for the measurement, calibration and diagnostic tool CANape. The “X” in XCP is a placeholder for different transport layers. For example, there are XCP-on-CAN, XCP-on-Ethernet, etc. and since February 2006 XCP-on-FlexRay as well. This is a single-master/multiple-slave concept, in which it is possible to communicate with ECUs very efficiently and with variable bandwidth for measurement and calibration purposes. While the slave can be integrated in the FlexRay stack, the tool itself provides master support for the protocol. The bandwidth is reallocated to individual nodes at runtime as needed. An enhanced version of the FlexRay driver is necessary to implement buffer reconfiguration with XCP-on-FlexRay. This too exhibits flexible handling of components.

Just how buffer organization affects the overall system was described by Dr. (Engineering) Mathias Rausch, editor of the FlexRay protocol specification, who is responsible for the FlexRay-related issues at Freescale. Rausch described in detail ways to optimize buffer loading in configuring different static or dynamic segments. In addition, Freescale takes advantage of the fact that the Controller Host Interface (CHI) in the FlexRay protocol is not specified in detail; only minimum requirements are specified as constraints. This gives the semiconductor producer the freedom to offer special functions. Optimal design of the CHI contributes toward making the software easier to structure at a later time. Rausch recommended the use of tools, because “in configuring up to 128 message buffers a lot of parameters need to be considered.”

Patrick Heuts, director of innovation and development management in the automotive business area at NXP Semiconductors, sought out more economical FlexRay components at the request of Dr. Schedl. “Besides transceivers, we also offer a FlexRay controller that is fully integrated in a microcontroller as a single-chip solution. Compared to a FlexRay controller that is normally embedded in the microcontroller as a peripheral device, this fully integrated solution offers decisive advantages. For example, the number and type of message buffers can be configured flexibly. The microcontroller has optimal and quick access to data in the local memory of the MCU. Actually, the fully integrated FlexRay controller operates more like a DMA engine with one or two channels. The next step for NXP Semiconductors will be to investigate whether integration of the transceiver in the chip might further minimize the costs of a system solution”, Figure 5.



[Figure 5: NXP Semiconductors offers the "MCU Centric" solution with its advantages of full integration of the FlexRay controller in the microcontroller.]

Although one of the stated goals is 'cost reduction' - a FlexRay system is not any more expensive than a comparable CAN architecture. Chip costs for FlexRay are higher than for CAN due to the extra silicon area that is needed, but according to internal studies at BMW, the total system costs are very comparable yet achieve higher performance, expandability and lower complexity.

Summary

FlexRay has much potential. The pilot project at BMW is only the beginning, and it has demonstrated that a FlexRay system - once set up properly - can run with stability. Universal tool support is absolutely recommended for the various development stages in order to maintain a clear perspective of the many different requirements. Tools with

extensive checking systems simplify the work of developers and help prevent errors from the outset. A large number of computationally and communicatively intensive applications can soon become reality thanks to FlexRay.

Revised: 6/2007

Word count: 1,829

Character count: 11,956

Figures:

Figure 1: Fixed allocation of transmission slots to individual components (A, B, C) simplifies their incorporation in system integration.

Figure 2: The state diagram of a schedule table shows how synchronization is achieved. The schedule is started, but it need not be absolutely synchronous right away (RUNNING). To run synchronously (AND SYNCHRONOUS) it can be put in a wait state (SCHEDULETABLE_WAITING), or it can be soft synchronized.

Figure 3: In CANoe, actions can be executed regularly in reference to the beginning of a cycle or after a specific slot has ended. Naturally, notifications are also possible when frames are received or are missing on the bus.

Figure 4: Layout of the FlexRay software according to AUTOSAR enables simple adaptation to different requirements. The figure shows a FlexRay stack, with a driver that is extended relative to AUTOSAR and is supplemented by a XCP transport layer and protocol modules.

Figure 5: NXP Semiconductors offers the "MCU Centric" solution with its advantages of full integration of the FlexRay controller in the microcontroller.

Figure 1: BMW AG

Figures 2-4: Vector Informatik GmbH

Figure 5: NXP Semiconductors

Literature references:

[1] Mayer, E.: Datenkommunikation im Automobil. Teil 4: FlexRay für den Datenaustausch in sicherheitskritischen Anwendungen ["Data communication in the automobile. Part 4: FlexRay for data exchange in safety-critical applications"]. Elektronik Automotive, 2007, Issue 2, pp. 42ff.

Author:



Dr. (rer. nat.) Carsten Böke studied Informatics at the University of Paderborn. From 1995 to 2004 he was a member of the scientific staff in the area of Design of Parallel Systems at the Heinz Nixdorf Institute. There he was primarily involved with automatic configuration of embedded communication systems. Since 2004 he has been employed as Senior Software Development Engineer at Vector Informatik GmbH where he develops tools for bus analysis and bus simulation of FlexRay systems.
Tel. +49-711/80670-4923, Fax +49-711/80670-111,
E-mail: carsten.boeke@vector-informatik.de

Vector Informatik GmbH
Ingersheimer Str. 24
70499 Stuttgart
Germany
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