Document Type: Tutorial
NI Supported: Yes
Publish Date: Aug 21, 2009

FlexRay Automotive Communication Bus Overview

Overview

The FlexRay communications bus is a deterministic, fault-tolerant and high-speed bus system developed in conjunction with automobile manufacturers and leading suppliers. FlexRay delivers the error tolerance and time-determinism performance requirements for x-by-wire applications (i.e. drive-by-wire, steer-by-wire, brake-by-wire, etc.). This article covers the basics FlexRay.

Table of Contents

1. Increasing Communications Demands
2. FlexRay Basics
3. FlexRay Topology and Layout
4. The FlexRay Protocol
5. FIBEX - The FlexRay network database
6. PCI and PXI FlexRay interfaces
7. Conclusion

Increasing Communications Demands

For automobiles to continue to improve safety, increase performance, reduce environmental impact, and enhance comfort, the speed, quantity and reliability of data communicated between a car's electronic control units (ECU) must increase. Advanced control and safety systems--combining multiple sensors, actuators and electronic control units--are beginning to require synchronization and performance past what the existing standard, Controller Area Network (CAN), can provide. Coupled with growing bandwidth requirements with today's advanced vehicles utilize over five separate CAN busses, automotive engineers are demanding a next-generation, embedded network. After years of partnership with OEMs, tool suppliers, and end users, the FlexRay standard has emerged as the in-vehicle communications bus to meet these new challenges in the next generation of vehicles.

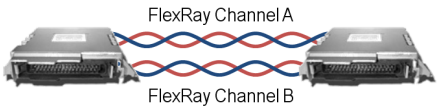
Adoption of a new networking standard in complex embedded designs like automobiles takes time. While FlexRay will be solving current high-end and future mainstream in-vehicle network challenges, it will not displace the other two dominant in-vehicle standards, CAN, and LIN. In order to optimize cost and reduce transition challenges, the next generation of automobiles will contain FlexRay for high-end applications, CAN for mainstream powertrain communications and LIN for low-cost body electronics.

Bus	LIN	CAN	FlexRay
Speed	40 kbit/s	1 Mbit/s	10 Mbit/s
Cost	$	$$	$$$
Wires	1	2	2 or 4
Typical Applications	Body Electronics (Mirrors, Power Seats, Accesories)	Powertrain (Engine, Transmission, ABS)	High-Performance Powertrain, Safety (Drive-by-wire, active suspension, adaptive cruise control)

Understanding how FlexRay works is important to engineers across all aspects of the vehicle design and production process. This article will explain the core concepts of FlexRay.

FlexRay Basics

Many aspects of FlexRay are designed to keep costs down while delivering top performance in a rugged environment. FlexRay uses **unshielded twisted pair** cabling to connect nodes together. FlexRay supports single- and dual-channel configurations which consist of one or two pairs of wires respectively. Differential signaling on each pair of wires reduces the effects of external noise on the network without expensive shielding. Most FlexRay nodes typically also have power and ground wires available to power transceivers and microprocessors.



Dual-channel configurations offer enhanced fault-tolerance and/or increased bandwidth. Most first-generation FlexRay networks only utilize one channel to keep wiring costs down, but as applications increase in complexity and safety requirements, future networks will use both channels.

FlexRay buses require **termination** at the ends, in the form of a resistor connected between the pair of signal wires. Only the end nodes on a multi-drop bus need termination. Too much or too little termination can break a FlexRay network. While specific network implementations vary, typical FlexRay networks have a cabling impedance between 80 and 110 ohms, and the end nodes are terminated to match this impedance. Termination is one of the most frequent causes of frustration when connecting a FlexRay node to a test setup. Modern PC-based FlexRay interfaces may contain on-board termination resistors to simplify wiring.

FlexRay Topology and Layout

One of the things that distinguishes FlexRay, CAN and LIN from more traditional networks such as ethernet is its topology, or network layout. FlexRay supports simple multi-drop passive connections as well as active star connections for more complex networks. Depending a vehicle's layout and level of FlexRay usage, selecting the right topology helps designers optimize cost, performance, and reliability for a given design.

Multi-drop Bus

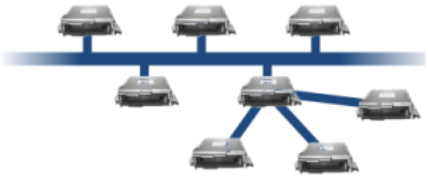
FlexRay is commonly used in a simple **multi-drop bus** topology that features a single network cable run that connects multiple ECUs together. This is the same topology used by CAN and LIN and is familiar to OEMs, making it a popular topology in first-generation FlexRay vehicles. Each ECU can "branch" up to a small distance from the core "trunk" of the bus. The ends of the network have termination resistors installed that eliminate problems with signal reflections. Because FlexRay operates at high frequencies, up to 10 Mbit/s compared to CAN's 1 Mbit, FlexRay designers much take care to correctly terminate and lay out networks to avoid signal integrity problems. The multi-drop format also fits nicely with vehicle harnesses that commonly share a similar type of layout, simplifying installation and reducing wiring throughout the vehicle.

Star Network

The FlexRay standard supports "Star" configurations which consist of individual links that connect to a central active node. This node is functionally similar to a hub found in PC ethernet networks. The active star configuration makes it possible to run FlexRay networks over longer distances or to segment the network in such a way that makes it more reliable should a portion of the network fail. If one of the branches of the star is cut or shorted, the other legs continuing functioning. Since long runs of wires tend to conduct more environmental noise such as electromagnetic emissions from large electric motors, using multiple legs reduces the amount of exposed wire for a segment and can help increase noise immunity.

Hybrid Network

The bus and star topologies can be combined to form a **hybrid** topology. Future FlexRay networks will likely consist of hybrid networks to take advantage of the ease-of-use and cost advantages of the bus topology while applying the performance and reliability of star networks where needed in a vehicle.



The FlexRay Protocol

The FlexRay protocol is a unique time-triggered protocol that provides options for deterministic data that arrives in a predictable time frame (down to the microsecond) as well as CAN-like dynamic event-driven data to handle a large variety of frames. FlexRay accomplishes this hybrid of core static frames and dynamic frames with a pre-set **communication cycle** that provides a pre-defined space for static and dynamic data. This space is configured with the network by the network designer. While CAN nodes only needed to know the correct baud rate to communicate, nodes on a FlexRay network must know how all the pieces of the network are configured in order to communicate.

As with any multi-drop bus, only one node can electrically write data to the bus at a time. If two nodes were to write at the same time, you end up with contention on the bus and data becomes corrupt. There are a variety of schemes used to prevent contention on a bus. CAN, for example, used an arbitration scheme where nodes will yield to other nodes if they see a message with higher priority being sent on a bus. While flexible and easy to expand, this technique does not allow for very high data rates and cannot guarantee timely delivery of data. FlexRay manages multiple nodes with a **Time Division Multiple Access** or TDMA scheme. Every FlexRay node is synchronized to the same clock, and each nodes waits for its turn to write on the bus. Because the timing is consistent in a TDMA scheme, FlexRay is able to guarantee **determinism** or the consistency of data deliver to nodes on the network. This provides many advantages for systems that depend on up-to-date data between nodes.

Embedded networks are different from PC-based networks in that they have a closed configuration and do not change once they are assembled in the production product. This eliminates the need for additional mechanisms to automatically discover and configure devices at run-time, much like a PC does when joining a new wired or wireless network. By designing network configurations ahead of time, network designers save significant cost and increase reliability of the network.

For a TDMA network such as FlexRay to work correctly, all nodes must be configured correctly. The FlexRay standard is adaptable to many different types of networks and allows network designers to make tradeoffs between network update speeds, deterministic data volume, and dynamic data volume among other parameters. Every FlexRay network may be different, so each node must be programmed with correct network parameters before it can participate on the bus.

To facilitate maintaining network configurations between nodes, FlexRay committee standardized a format for the storage and transfer of these parameters in the engineering process. The Field Bus Exchange Format, or **FIBEX** file is an ASAM-defined standard that allows network designers, prototypers, validaters, and testers to easily share network parameters and quickly configure ECUs, test tools, hardware-in-the-loop simulation systems, and so on for easy access to the bus.

The Communication Cycle

The **FlexRay communication cycle** is the fundamental element of the media-access scheme within FlexRay. The duration of a cycle is fixed when the network is designed, but is typically around 1-5 ms. There are four main parts to a communication cycle:

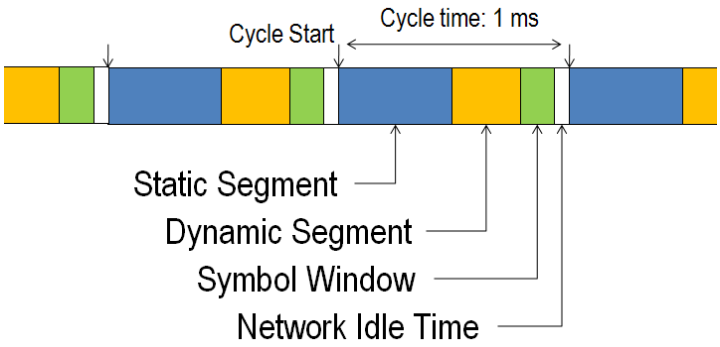


Figure 1: Communication Cycle

1. **Static Segment**
   Reserved slots for deterministic data that arrives at a fixed period.
2. **Dynamic Segment**
   The dynamic segment behaves in a fashion similar to CAN and is used for a wider variety of event-based data that does not require determinism.
3. **Symbol Window**
   Typically used for network maintenance and signaling for starting the network.
4. **Network Idle Time**
   A known "quiet" time used to maintain synchronization between node clocks.

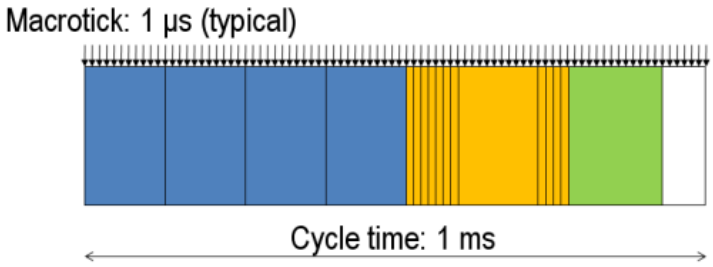


Figure 2. Detail of the FlexRay macrotick

The smallest practical unit of time on a FlexRay network is a **macrotick**. FlexRay controllers actively synchronize themselves and adjust their local clocks so that the macrotick occurs at the same point in time on every node across the network. While configurable for a particular network, macroticks are often 1 microsecond long. Because the macrotick is synchronized, data that relies on it is also synchronized.

1. Static Segment

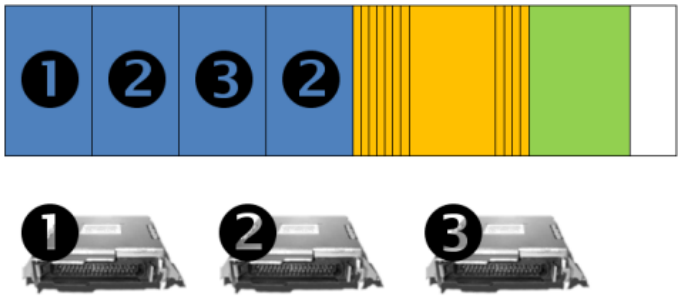


Figure 3: Illustration of a static segment with 3 ECUs transmitting data to 4 reserved slots.

The static segment, represented as the blue portion of the frame, is the space in the cycle dedicated to scheduling a number of time-triggered frames. The segment is broken up into slots, each slot containing a reserved frame of data. When each slot occurs in time, the reserved ECU has the opportunity to transmit its data into that slot. Once that time passes, the ECU must wait until the next cycle to transmit its data in that slot. Because the exact point in time is known in the cycle, the data is deterministic and programs know exactly how old the data is. This is extremely useful when calculating control loops that depend on consistently spaced data. **Figure 3** illustrates a simple network with four static slots being used by three ECUs. Actual FlexRay networks may contain up to several dozen static slots.

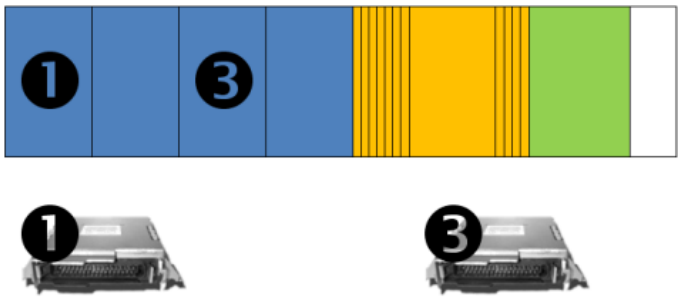


Figure 4. Illustration of a static slot with ECU #2 missing.

If an ECU goes offline or decides not to transmit data, its slot remains open and is not used by any other ECU, as shown in Figure 4.

2. Dynamic Segment

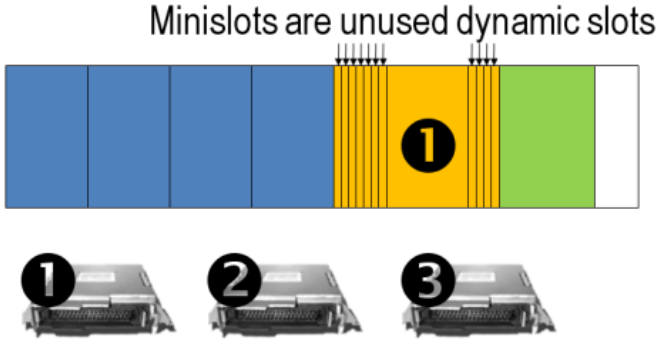


Figure 5. Illustration of FlexRay dynamic slots with one ECU broadcasting data.

Most embedded networks have a small number of high-speed messages and a large number of lower-speed, less-critical networks. To accommodate a wide variety of data without slowing down the FlexRay cycle with an excessive number of static slots, the dynamic segment allows occasionally transmitted data. The segment is a fixed length, so there is a limit of the fixed amount of data

that can be placed in the dynamic segment per cycle. To prioritize the data, **minislots** are pre-assigned to each frame of data that is eligible for transmission in the dynamic segment. A minislot is typically a **macrotick** (a microsecond) long. Higher priority data receives a minislot closer to the beginning of the dynamic frame.

Once a minislot occurs, an ECU has a brief opportunity to broadcast its frame. If it doesn't broadcast, it loses its spot in the dynamic frame and the next minislot occurs. This process moves down the minislots until an ECU elects to broadcast data. As the data is broadcast, future minislots must wait until the ECU completes its data broadcast. If the dynamic frame window ends, then the lower-priority minislots must wait until the next cycle for another opportunity to broadcast.

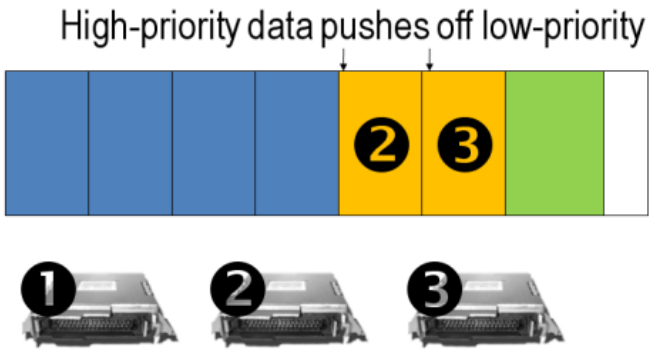


Figure 6. Dynamic slots illustration showing ECUs 2 and 3 broadcasting in their minislots and leaving no time for the lower-priority minislots.

Figure 5 shows ECU #1 broadcasting in its minislot since the first 7 minislots chose not to broadcast. Figure 6 shows ECUs #2 and #3 using the first two minislots, leaving no time for ECU #1 to broadcast. ECU #1 must wait for the next cycle to broadcast.

The end result of the dynamic segment is a scheme similar to the arbitration scheme used by CAN.

3. Symbol Window

The Symbol window is primarily used for maintenance and identification of special cycles such as cold-start cycles. Most high-level applications do not interact with the symbol window.

4. Network Idle Time

The network idle time is of a pre-defined, known length by ECUs. The ECUs make use of this idle time to make adjustments for any drift that may have occurred during the previous cycle.

Data Security and Error Handling

The FlexRay network provides scalable fault-tolerance by allowing single or dual-channel communication. For security-critical applications, the devices connected to the bus may use both channels for transferring data. However, it is also possible to connect only one channel when redundancy is not needed, or to increase the bandwidth by using both channels for transferring non-redundant data.

Within the physical layer, FlexRay provides fast error detection and signaling, as well as error containment through an independent Bus Guardian. The Bus Guardian is a mechanism on the physical layer that protects a channel from interference caused by communication that is not aligned with the cluster's communication schedule.

Frame Format

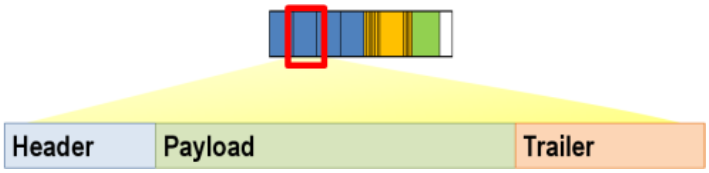


Figure 7. Detail of a FlexRay Frame
Each slot of a static or dynamic segment contains a FlexRay Frame. The frame is divided into three segments: Header, Payload, and Trailer.

Header

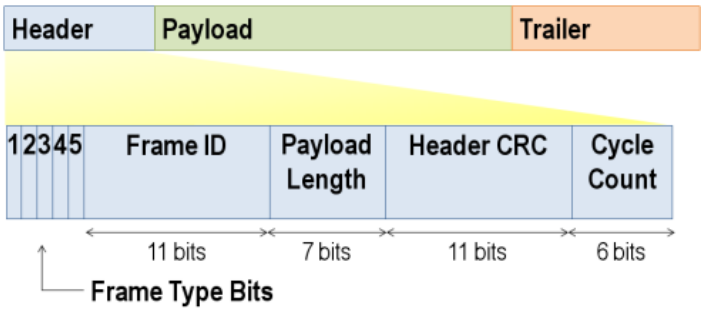


Figure 8. Bit-level breakdown of a FlexRay Frame

The Header is 5 bytes (40 bits) long and includes the following fields:

1. Status Bits - 5 bits
2. Frame ID - 11 bits
3. Payload Length - 7 bits

4. Header CRC - 11 bits
5. Cycle Count - 6 bits

The Frame ID defines the slot in which the frame should be transmitted and is used for prioritizing event-triggered frames. The Payload Length contains the number of words which are transferred in the frame. The Header CRC is used to detect errors during the transfer. The Cycle Count contains the value of a counter that advances incrementally each time a Communication Cycle starts.

Payload

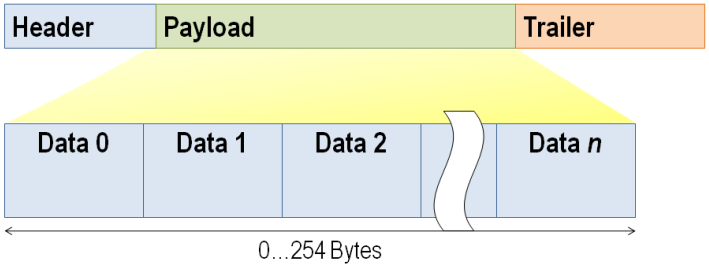


Figure 9. Payload of a FlexRay Frame.

The payload contains the actual data transferred by the frame. The length of the FlexRay payload or data frame is up to 127 words (254 bytes), which is over 30 times greater compared to CAN.

Trailer

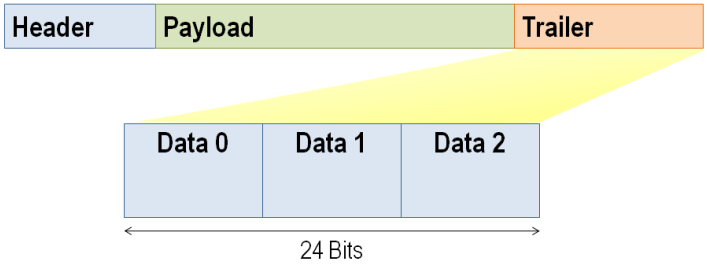


Figure 10. Trailer of a FlexRay Frame.

The trailer contains three 8-bit CRCs to detect errors.

Signals

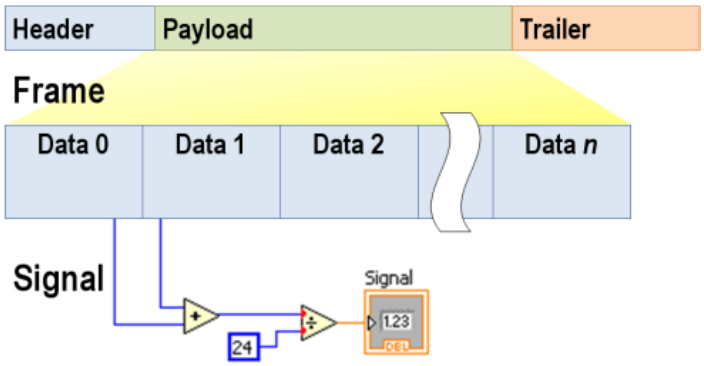


Figure 11. Frame to Signal conversion

FlexRay data is represented in bytes. Most applications require data to be represented in real decimal values with units, scaling, and limits. When you take one or more bits or bytes from a FlexRay frame, apply a scaling and offset, you get a **signal** that is useful for communicating actual parameters between ECUs. Most ECUs programs work with FlexRay data as signals and leave the conversion of signals to raw frame data up to the driver or lower-level communication protocols.

A typical vehicle has hundreds to thousands of signals. As the scaling, offset, definitions, and locations of these signals can change, FlexRay networks store these definitions in the FIBEX database that defines the network. This makes writing programs for FlexRay networks easier as designers can simply refer to the signal name in the code. The compiler or driver then pulls the most recent scaling and offset information when the program is updated to the ECU or test system.

Clock synchronization and cold starting

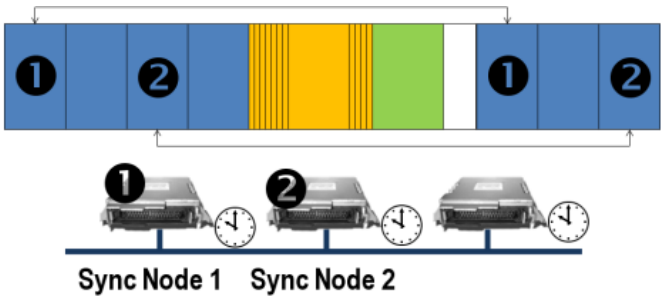


Figure 12. Simplified Synchronization process of a FlexRay network

FlexRay has the unique ability to sync up nodes on a network without an external synchronization clock signal. To do so, it uses 2 special types of frames: **Startup Frames** and **Sync Frames**. To start a FlexRay cluster, at least 2 different nodes are required to send startup frames. The action of starting up the FlexRay bus is known as a **cold-start** and the nodes sending the startup frames are usually known as cold-start nodes. The startup frames are analogous to a start trigger, which tells all the nodes on the network to start.

Once the network is started, all nodes must synchronize their internal oscillators to the network's macrotick. This can be done using two more more synchronization nodes. These can be any two separate nodes on the network that pre-designated to broadcast special sync frames when they are first turned on. Other nodes on the network wait for the sync frames to be broadcast, and measure the time between successive broadcasts in order to calibrate their internal clocks to the FlexRay time. The sync frames are designated in the FIBEX configuration for the network.

Once the network is synchronized and on-line, the network idle time (white space in the diagram) is measured and used to adjust the clocks from cycle-to-cycle to maintain tight synchronization.

In-cycle control

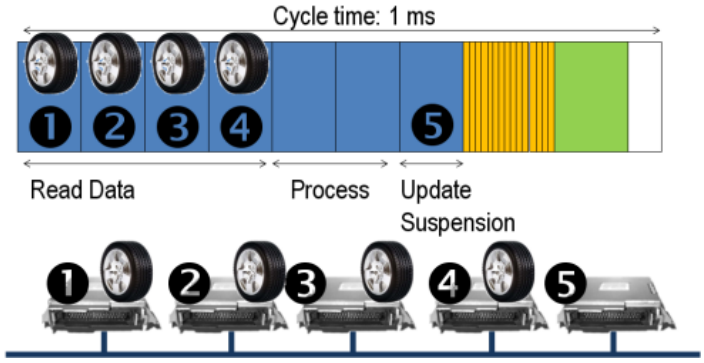


Figure 13. In-cycle control reading 4 wheel positions and updating a vehicle control output in a single FlexRay cycle.

An advanced feature of FlexRay is the ability to do in-cycle control. Figure 13 illustrates an example where four wheel positions are broadcast in the static slots of the frame. Because the wheel positions occur before the final update command from the central controller #5, the controller has time to process and make a rapid output within the same communication cycle. This allows very high-speed control rates to be realized on a FlexRay network.

FIBEX - The FlexRay network database

The FIeld Bus EXchange (FIBEX) format, an XML-based standardized file format defined by the ASAM consortium, is used for describing automotive networks. While the standard format for FlexRay networks, the FIBEX database format is compatible with many different automotive protocols, making it a flexible standard. FIBEX databases are typically generated by vehicle network designers and shared with engineers working on a particular aspect of the vehicle. With a FIBEX file and an PC interface or ECU that supports it, you can easily interact with a vehicle network without having to manually configure interfaces and signal definitions.

FIBEX contains many aspects for a particular network, including the following:

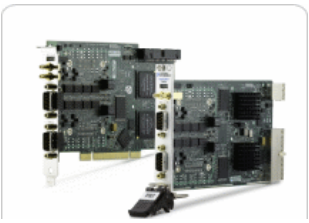
- Transmit and receive schedules
- Frame definitions
- Signal definitions
- Bit-level encoding of signals
- Network topology
- ECU information
- Network configurations, including baud rates and timings

For more information on FIBEX, see the Introduction to FIBEX article,

PCI and PXI FlexRay interfaces

National Instruments offers high-performance PCI and PXI FlexRay interfaces for connecting PCs to FlexRay networks. With a PC-based interface, you can perform many engineering tasks on a FlexRay-enabled ECU, including:

- Rapid Prototyping
- Hardware-in-the-loop simulation
- Bus logging and debugging
- Signal data acquisition
- System Diagnostics
- Custom applications



For more information on NI FlexRay interfaces, see the Introduction to the NI-XNET FlexRay platform.

Conclusion

The FlexRay communications network delivers the deterministic, fault-tolerant and high-speed bus system performance requirements for the next generation of automobiles.

Related Links:

- Introduction to FIBEX
- Introduction to the NI-XNET FlexRay platform.

Legal
This tutorial (this "tutorial") was developed by National Instruments ("NI"). Although technical support of this tutorial may be made available by National Instruments, the content in this tutorial may not be completely tested and verified, and NI does not guarantee its quality in any way or that NI will continue to support this content with each new revision of related products and drivers. THIS TUTORIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND AND SUBJECT TO CERTAIN RESTRICTIONS AS MORE SPECIFICALLY SET FORTH IN NI.COM'S TERMS OF USE (http://ni.com/legal/termsofuse/unitedstates/us/).