Networking the FlexRay Way

An overview of the FlexRay Communications System

Dr. Christopher Temple, Freescale Semiconductor, Strategy and Advanced Systems Lab

Driven by the ongoing shift from mechanical to electrical systems in vehicles the FlexRay consortium is defining a communications system that encompasses the specification of a communications protocol as well as the specification of key needs of the physical media for next-generation distributed automotive control.

Placing a focus on the communications protocol and the physical media gives FlexRay a well defined scope within the automotive E/E architecture and provides the openness to be incorporated in higher level frameworks that define the overall automotive E/E architecture from a top-down perspective, such as AUTOSAR. FlexRay is focused around a set of core needs which will now be outlined.

Data transfer rate and network topology needs

When the consortium set out it became obvious that the existing data transfer rates used in today's vehicles within the body, chassis and powertrain domain will reach their limit in the next-generation systems. The consortium selected a data transfer rate of 10MBit/s as a viable compromise that balances the need for speed against the techno-economical constraints that always require special consideration in automotive systems. Currently support for additional lower data transfer rates is under consideration.

For interconnection the consortium decided to primarily consider two types of topologies. These are a star-based interconnection topology and a bus-based interconnection topology. In both cases, duplication of the interconnection is optional, i.e. FlexRay can operate not only as a single channel system such as CAN and LIN, but also as a dual channel system. The star configuration of FlexRay can also be deployed in distributed configurations such that two star-based subsystems are connected by star-to-star links. A distributed system of FlexRay nodes can be also designed by combining the star and the bus approach allowing several nodes to be connected to a branch of the star.

Data communication needs

Within FlexRay communication occurs in recurring communication cycles. Each communication cycle is built on a four level timing hierarchy as depicted in Figure 1. In order to satisfy diverse communication requirements FlexRay provides a static communication segment and a dynamic communication segment within each communication cycle. In addition each cycle contains a symbol window that is used for run-time testing, and the network idle time which is a communication-free period that concludes each communication cycle.

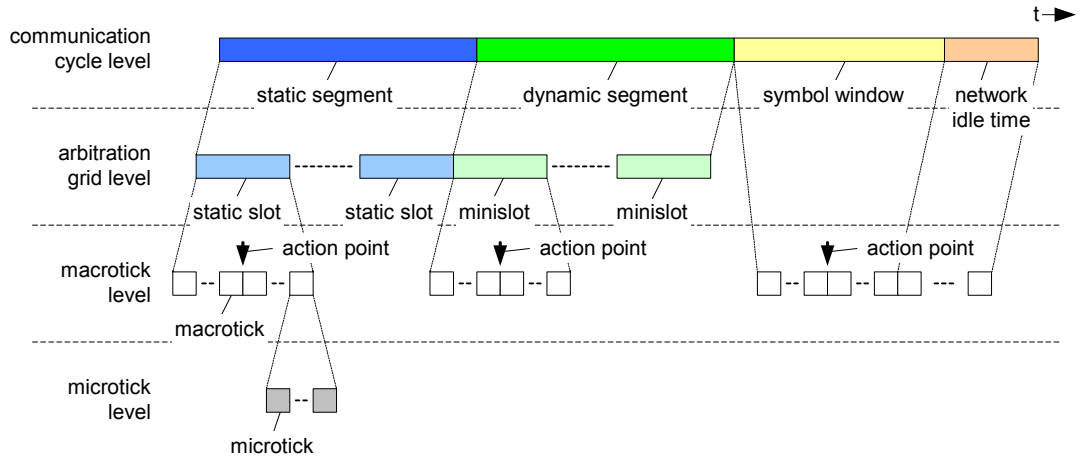


Figure 1: FlexRay timing hierarchy.

The static communication segment addresses data communication that requires bounded latency and small latency jitter. In order to achieve this, the static communication segment uses a TDMA-based communication scheme. In combination with a static schedule that is calculated off-line during the design of the system FlexRay provides a viable infrastructure for running highly deterministic distributed applications, such as closed-loop control applications where the control loop is closed over the network. The dynamic segment addresses ad-hoc data communication needs that imply varying bandwidth requirements that may emerge at run-time of the system, such as for the transfer of diagnosis data. Within the dynamic communication segment different nodes may compete for bandwidth using a priority-driven mini-slotting scheme.

The temporal characteristics of the communication cycle are defined at design time and stored statically in each node. Nodes that require greater bandwidth and those that need shorter update intervals for messages are assigned more slots than those that require less bandwidth or those that allow for longer up-date intervals for messages.

To facilitate the synchronous execution of the communication cycle each network node contains a time base that generates synchronized macroticks. The time bases are synchronized such that all macroticks are kept in synchronization with one another. The macroticks are formed out of multiple unsynchronized microticks that are generated, for example, by an oscillator local to the node. This synchronization is performed autonomously by the communications protocol. Apart from driving the communication cycle the synchronized time base is also made available to the application to enable synchronized operation of the overall system.

Fault-tolerance needs

Adequately addressing fault-tolerance is one of the key aspects that needed to be considered during the design of FlexRay. To allow a single communications system to support the diverse needs of automotive applications across different application domains the consortium decided to introduce a concept of scalable fault-tolerance. Scalable fault-tolerance aims at allowing FlexRay to be used economically in distributed non fault-tolerant systems as well as in distributed fault-tolerant systems.

To support the concept of scalable fault-tolerance various provisions have been made. First, FlexRay supports different interconnection topologies: a designer may choose to operate FlexRay as a single-channel system, as a dual-channel system and as a dual-channel system with mixed connectivity, where some nodes are connected to both channels while other nodes are connected to only one. In addition FlexRay can be deployed using optional local or remote channel guardians that protect the communications channels from transmission faults that violate the TDMA scheme. Second, the clock synchronization algorithm supports fault-tolerant as well as non fault-tolerant synchronization. For fault-tolerant synchronization the synchronization algorithm considers the transient / permanent fault class as well as the symmetric / asymmetric fault class. Last, but not least FlexRay adheres to a separation of the functional and the structural domain described in the following.

Conceptual partitioning needs

The consortium realized early on that it would be highly beneficial for FlexRay to support a clean separation between the functional domain and the structural domain of the automotive E/E architecture. As illustrated above the temporal characteristics of the communication cycle are defined at design time and stored statically in each node. The nodes are not, however, provided with the overall communication matrix and, in particular, the communications protocol does not require knowledge of the physical identity of senders of messages.

This strategy allows functions that are spread across multiple nodes to be integrated into a single node and functions that are provided by a single node to be separated into multiple nodes after deployment of the system, in a way, that is fully transparent to the remaining nodes in the system. This is beneficial for migrating nodes across different clusters that have a different functional partitioning within the cluster.

Care was also taken to ensure that the communications protocol focuses on communication-related fault-tolerance needs but not on application-related ones such as application related message agreement algorithms, for example. This design paradigm ensures that FlexRay can be deployed economically across a wide set of different applications that have specific but different application-related fault-tolerance needs.

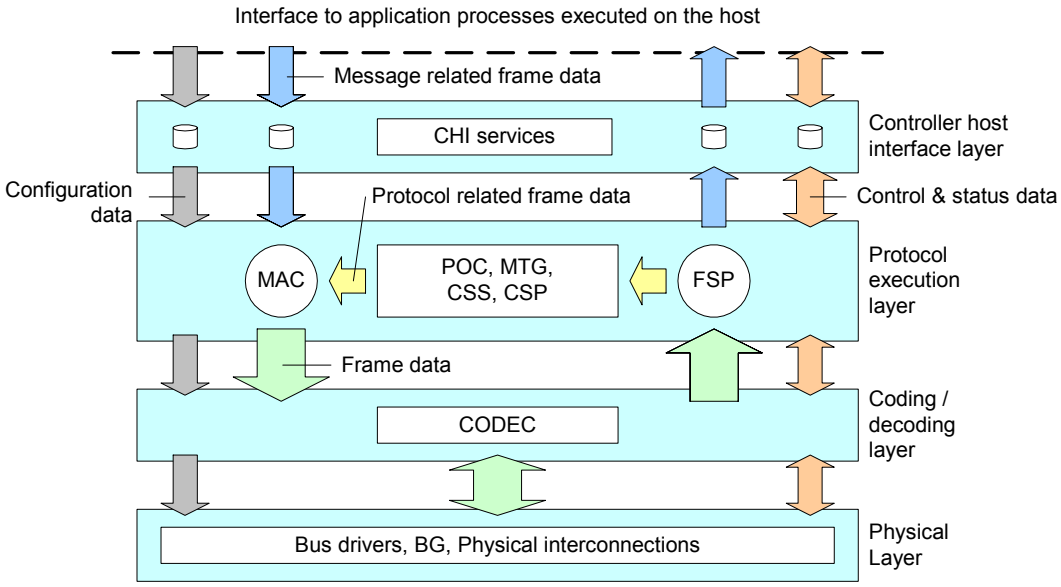
Host interface service needs

FlexRay also includes services at the host interface. These services are located in the controller host interface layer as they are transparent to the underlying communications protocol layers.

Functions currently defined include a timer service, an interrupt service, a message ID filtering service, and a network management service. The timer service allows time-outs to be defined for the application based on the synchronized time base maintained by the protocol. Upon reaching a time-out the interrupt service raises an interrupt to the host. The message ID filtering service provides means for selecting receive buffers based on a message ID that is exchanged in frames that are enabled for message ID filtering. This allows using a message selection concept such as in CAN, where message IDs embed the semantical meaning of the associated message. The network management service supports an application-level functional network management strategy, where functions may sign on or off to other functions within the system without knowledge of their physical location, i.e. without knowing which node actually performs the function. Since these controller host interface services are transparent to the underlying protocol layers it is possible to extend these services without breaking interoperability of the communications protocol at a later time.

Conceptual hierarchy

Figure 2 depicts a conceptual layer hierarchy of the FlexRay communications system.



CHI ... Controller host interface
CSP ... Clock sync processing
CSS ... Clock sync startup
FSP ... Frame / symbol processing
MAC ... Media access control
MTG ... Macrotick generation
POC ... Protocol operation control

Figure 2: Conceptual hierarchy of the communications system layers.

The central layer of this hierarchy is the protocol execution layer. Within the protocol execution layer outgoing frame data is sent to the physical layer according to the time-driven media access strategy defined for media access control (MAC). Frame data contains not only 0-254

bytes of message data that is held in the payload section of the frame, but also 5 bytes of protocol related data that is held in the header section of the frame. The frame is secured using a 24 bit CRC that is stored in the trailer segment of the frame.

Incoming frame data is checked by frame / symbol processing (FSP) against a set of syntactical and semantical acceptance criteria. The message contained in an accepted frame is passed to the controller host interface (CHI) for storage while the protocol related frame data is provided to the core processes of the protocol execution layer. These core processes consist of the protocol operation control (POC), the macrotick generation (MTG), the clock synchronization startup (CSS) and the ongoing clock synchronization processing (CSP) that executes the synchronization algorithm.

On the one hand the protocol execution layer interfaces to the controller host interface layer that contains storage means for all interface data and the controller host interface services, on the other hand the protocol execution layer interfaces to the coding / decoding layer that performs the non-return to zero (NRZ) coding and decoding of frames. The frames are exchanged among nodes on the physical layer, which forms the lowest level of the hierarchy. The physical layer contains the bus drivers, the bus guardians and the physical interconnections including any star couplers or other hubs that are located in the interconnection path.

Industry backing and development

In order to ensure success for the FlexRay communications system broad support throughout the industry is required. Having started back in 2000 by an agreement among four core members the FlexRay consortium has now grown to encompass 7 core members that are backed by a strong force of 11 premium associate members, 42 associate members and 26 development members, who have so far joined the consortium.

Within the consortium the work on FlexRay is performed in 9 working groups that are coordinated by the steering committee and the executive board. 6 of these working groups are aligned according to a V model and address requirements, protocol development, physical layer development, testing, protocol conformance testing, and physical layer conformance testing. In addition working groups are installed that focus on safety related aspects, FMEA, and the analysis of core communications system services.

The FlexRay specifications can be obtained along with further information from the FlexRay consortium web site at http://www.flexray.com.