Network Management and Transport Layer in TimeCore

Thomas M. Galla
DECOMSYS GmbH
Stumpergasse 48/28
A-1060 Wien
Tel: +43 (1) 59983 15
FAX: +43 (1) 59983 715
galla@decomsys.com

Jochen Olig
3SOFT GmbH
Frauenweiherstrasse 14
D-91058 Erlangen
Tel: +49 (9131) 7701 120
FAX: +49 (9131) 7701 333
Jochen.Olig@3SOFT.de

1 Introduction

During the last few years, major automotive companies have been striving for the deployment of standard software components since the potential benefits of standard software components are huge [1], a key motivation for the formation of the AUTOSAR consortium. Important issues in this context are safety (increased test depth of standard software components), software reuse, for the possibility to combine software components supplied by different vendors, and last but not least cost reasons in order to cope with ever reducing development cycles.

TimeCore [2], a joint product of 3SOFT GmbH and DECOMSYS GmbH, provides a well-integrated set of such standard software components. Apart from its major components, the time-driven operating system OSEKtime OS [3], the event-driven operating system OSEK/VDX OS [4], and the fault-tolerant communication layer OSEKtime FTCom [5], TimeCore features

- a transport layer for the transmission of data packets larger than maximum transfer unit (MTU) of the underlying communication system and
- a network management service taking care of coordinated mode changes of all ECUs connected to the communication network.

This article focuses on these latter two services, namely transport layer and network management.

2 FlexRay Transport Layer

In general, a transport layer facilitates the transmission of messages whose length is greater than the maximum transfer unit (MTU) of the underlying communication system. On the sender's side, the transport layer will split such long messages into multiple packets, which can be handled by the underlying communication system. On the receiver's side, the transport layer reassembles these packets again.

TimeCore provides implementations of this transport layer that are built on top of the CAN communication system and on top of the FlexRay communication system [6, 7]. Whereas the CAN transport layer is compliant to the ISO standard [8], the FlexRay implementation can only be close to this standard due to the lack of an appropriate standard for FlexRay.

2.1 Basic Operation

The FlexRay implementation of the TimeCore transport layer provides *acknowledged peer-to-peer data transfer* of up to 16 megabytes. The different frame types used by the transport layer are very similar to the ones used in the ISO transport layer for CAN.

If the amount of data to be sent is smaller than the maximum payload of a transport layer frame, only a so-called *single frame* is sent. If this transmission is performed successfully, the recipient answers with a flow *control frame* containing a *positive acknowledgment* (Figure 1(a)). In case the transmission of the single frame is faulty, the sender answers with a *flow control* containing a *negative acknowledgment* (Figure 1(b)).

In case the data to be sent is larger than the maximum payload of a transport layer frame, segmentation must take place at the sender side and reassembly at the recipient's side. In that case the sender transmits a so-called *first frame* (which contains the total amount of data bytes to be transferred). The receiver answers with a *flow control clear to send frame*. Triggered by this flow control clear to send frame, the sender transmits a block of up to 16 *consecutive frames*, each one containing a chunk of data and a unique sequence number. The exact amount of consecutive frames within a block is determined by the block size parameter in the flow control clear to send frame of the receiver. After a complete transmission of a block of consecutive frames, the recipient again has to trigger the next block of consecutive frames by a flow control clear to send frame. This procedure is repeated until all data has been transferred to the receiver. If this is the case the receiver acknowledges the complete transmission with a flow control frame containing a positive acknowledgement (Figure 1(d)). If some consecutive frame is not received correctly, the receiver has the possibility to request the retransmission of all consecutive frames contained in current block starting from the consecutive frame that has been incorrectly received by transmitting a flow control frame containing a negative acknowledgement and the sequence number of the first frame to retransmit (Figure 1(c)).

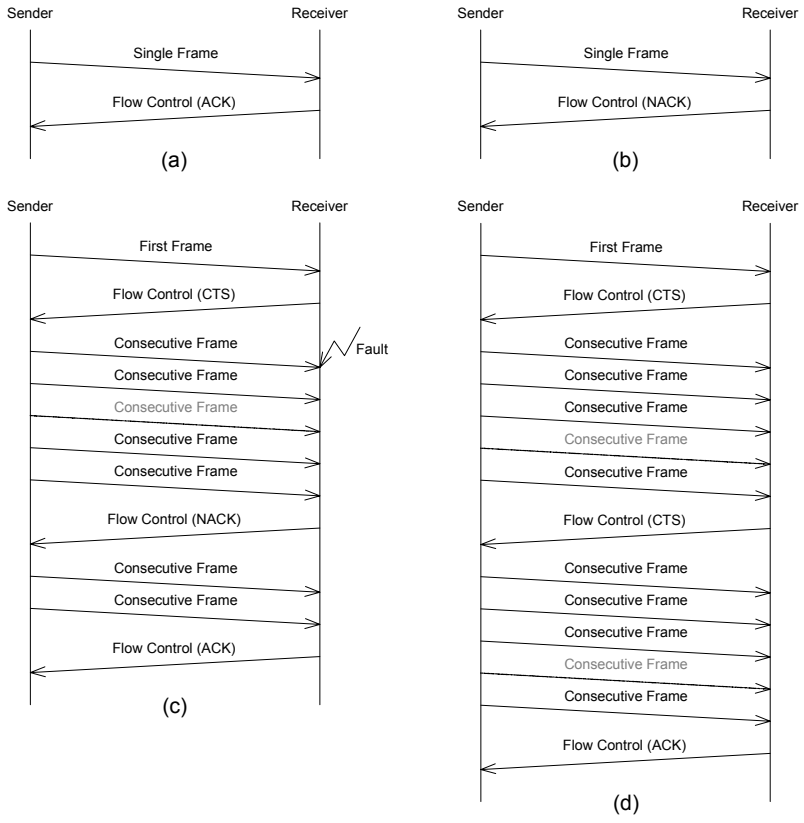

Figure 1: Segmented and Unsegmented Data Exchange

Different from the ISO transport layer for CAN, the FlexRay implementation of the TimeCore transport layer supports the handling of multiple parallel transport layer connections at the same time. – In that case each connection is uniquely identified by the 2-tuple made up of sender and the receiver address or by a unique session ID (Section 2.3).

2.2 Addressing Scheme

In order to address different ECU within a network in a unique fashion, each ECU is given a unique physical address (similar to the IP addresses). TimeCore's addressing scheme supports direct addressing (sending a message to a specific node), broadcast addressing (sending a message to all nodes in the network), and multicast addressing (sending a message to a specific subset of nodes). Furthermore, TimeCore's addressing scheme distinguishes between physical addressing (based on an ECU's physical address) and functional addressing (based on the functions performed by a specific ECU).

2.3 Frame Format

The different types of frames (as described in Section 2.1) used by the transport layer adhere to the frame format depicted in Figure 2. – The frame format is designed for MTUs of at least 32 bytes.

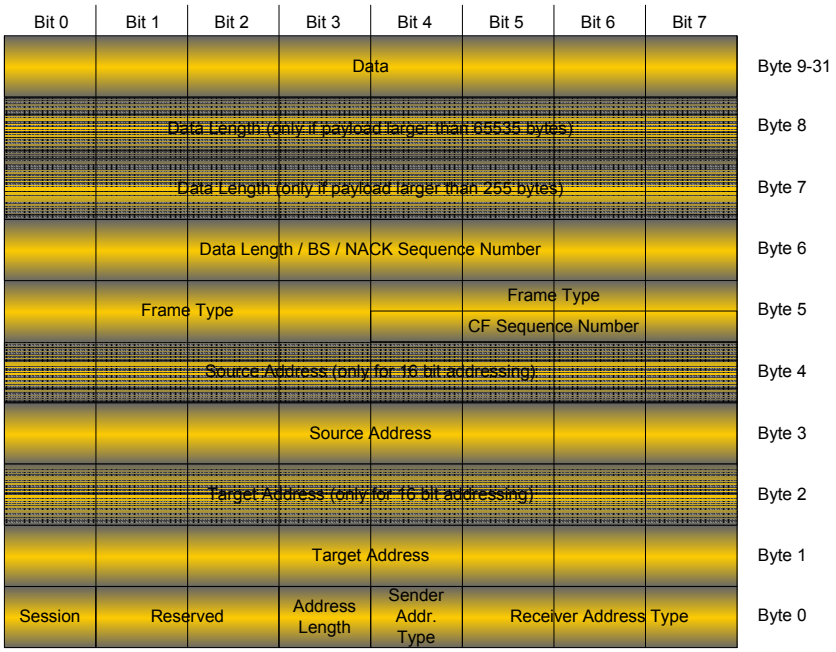

Figure 2: FlexRay Transport Layer Frame Format

Hereby byte 0 is a control byte that defines the layout of the remaining frame. The session bit defines whether a connection of the transport layer is identified by the 2-tuple of sender- and receiver address or by a unique session number. The address length bit indicates whether 8 or 16 bit addresses are used. The sender address type defines whether the source address identifies a single ECU (physical addressing) or a single function (functional addressing). Depending on the receiver address type bits, the semantics of the target address (single ECU, single function, group of ECUs, group of functions) is determined.

Byte 1 and 2 (in case of 16 bit addresses) contain the target address, whereas byte 3 and 4 (in case of 16 bit addresses) contain the source address.

Byte 5 determines the frame type (i.e., single frame, first frame, consecutive frame, or flow control frame) and contains the sequence number in case of consecutive frames.

Byte 6 contains the first byte of the data length in case of single- and first frames, the desired block size in case of flow control clear to send frames, and the sequence number in case of flow control negative acknowledge frames.

Byte 7 and 8 contain subsequent bytes of the data length depending on the actual amount of data bytes to be transmitted.

Using this frame format, the maximum payload length for a transport layer frame is 28 bytes in case of 32 byte MTUs.

2.4 FlexRay Schedule

The FlexRay implementation of the transport layer makes use of the dynamic part of FlexRay for the transmission of the transport layer frames. Hereby the bandwidth available to the transport layer can flexibly be chosen upon system design time. A contingent region of minislots within the dynamic part of FlexRay is reserved for the transport layer. This region is divided into three distinct sub regions as depicted in Figure 3. The setup depicted in Figure 3 assumes that the FlexRay network is connected via a dedicated master gateway to the main diagnostic network of the car (currently CAN). – Thus diagnostic access to the different ECUs connected to FlexRay or even flash download has to take place via this master gateway.

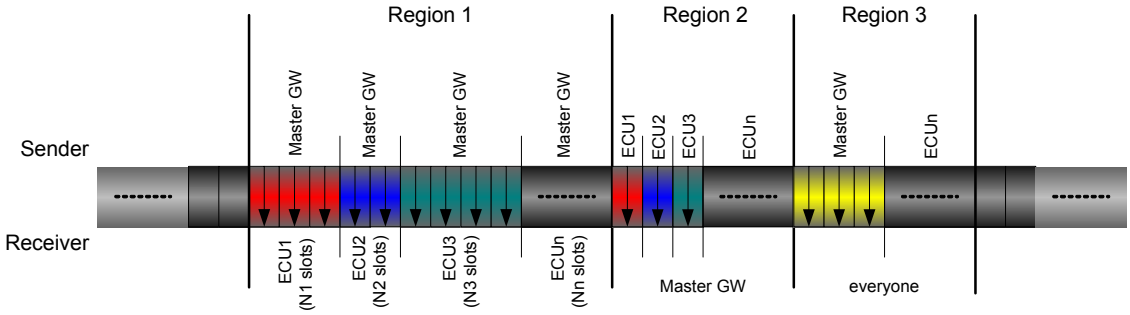

Figure 3: Transport Layer FlexRay Schedule

Region1 is reserved for the communication from the master gateway to the different ECUs. Hereby a dedicated number of minislots can be reserved for the communication with each ECUs, thus allowing for precise offline bandwidth allocation on a per ECU basis.

Region 2 is reserved for the communication from each ECU to the master gateway. Since the required bandwidth for this communication will be rather small (only diagnostic responses) a single minislot per ECU is reserved here.

Region 3 serves as a broadcast region, where the master gateway is able to communicate with multiple ECUs at the same time.

2.5 Configuration

The implementation of the transport layer consists of static code sections and configuration files. The configuration files contain the communication settings for the transport layer like allocated FlexRay buffers, number of send and receive slots for the master gateway and the involved ECUs and addressing information.

The transport layer set-up is configured for all nodes together using a dedicated configuration tool. This tool is implemented as plug-in of the Timeore toolchain. During execution it performs the following tasks:

- **Slot Assignment:** With the aid of the configuration tool a certain amount of minislots can be reserved for the transport layer within FlexRay's dynamic segment. From this pool of reserved minislots, a defined number can be assigned to each ECU (Figure 4).
- **Buffer Reservation:** For the previously defined minislots a pool of FlexRay buffers is reserved (Figure 4).

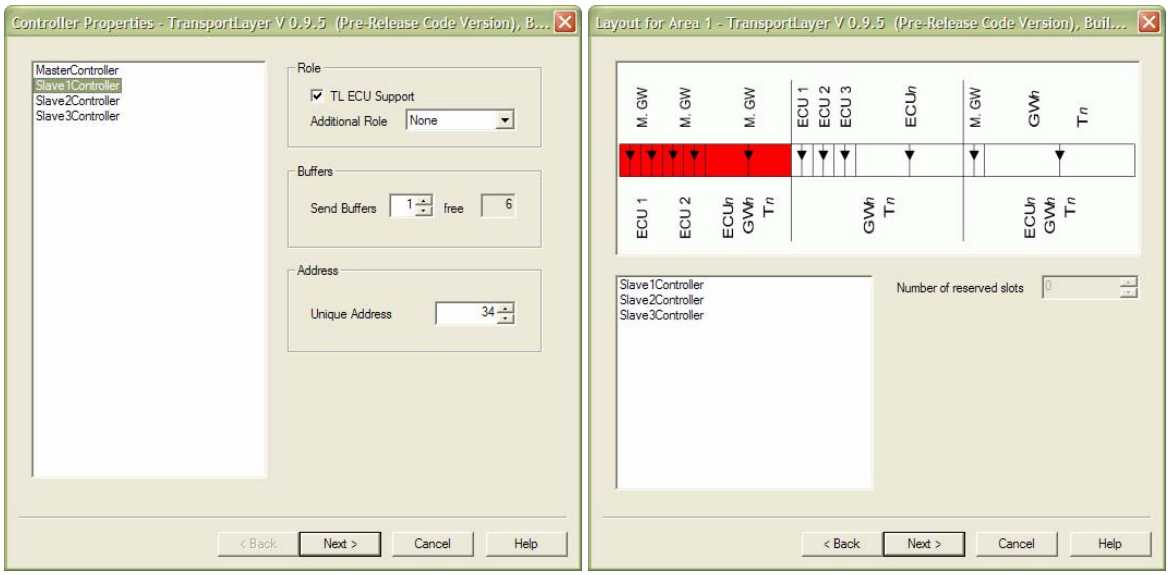

Figure 4: Transport Layer Configuration – Assigning Slot and Reserving FlexRay Buffers

- **Definition of Groups:** In this step address groups can be defined and ECUs can be defined to be members of these groups. In the left part of Figure 5 for example a group named "Wheel" is defined with the ECUs Slave1Controller, Slave2Controller, Slave3Controller as group members.
- **Generation of Configuration Files:** In the last step, the configuration tool produces a set of configuration files which contain the information entered in the previous three steps. – Hereby one file for each ECU is generated in the chosen directory (Figure 5).

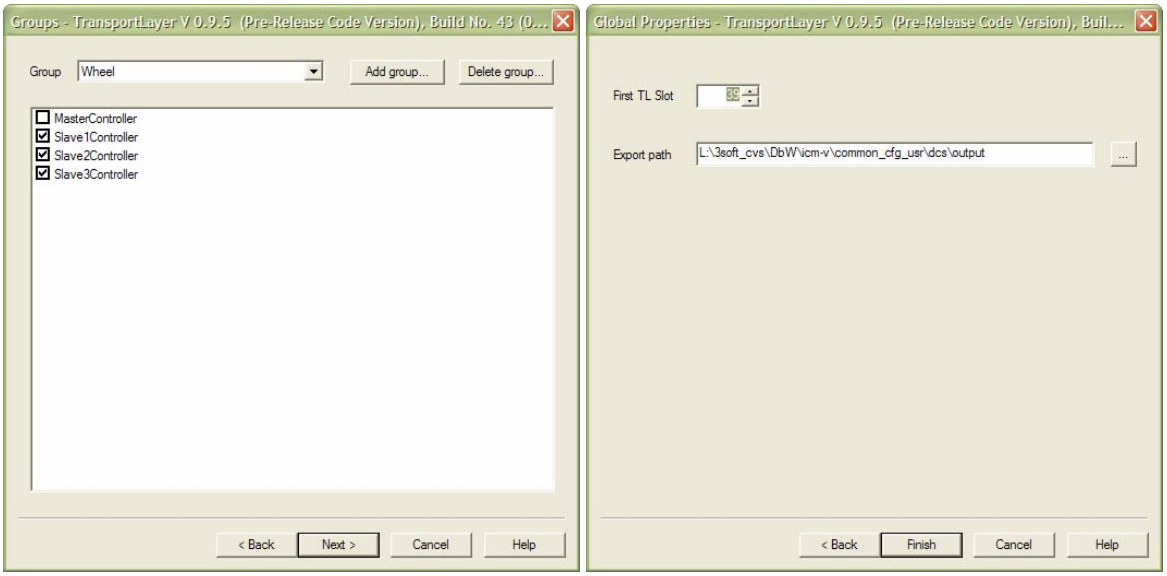

Figure 5: Transport Layer Configuration – Definition of Groups and Configuration File Generation

2.6 Higher Layer Protocols

Based on the services provided by the transport layer, higher-level services as well as diagnostic protocols can be implemented. Boot loading or flash programming services (as widely used by today's automotive manufacturers for performing maintenance updates in the field) are examples for such higher-level services. A representative for the field of diagnostic protocols is KWP2000 [9], an ISO standard which describes a set of functions with which an ECU can be diagnosed using a dedicated tester tool.

2.7 Comparison to ISO TP for CAN

When compared to the ISO transport layer for CAN, the FlexRay implementation for TimeCore provides several advantages:

- **Acknowledged Data Exchange:** Different from the ISO TP for CAN the FlexRay implementation provides acknowledgment to the sender of a transmission, whether or not the transmission has been successfully received by the recipient.
- **Partial Retransmission:** If some consecutive frames within a block of consecutive frames are not received correctly, the recipient can request retransmission of these frames.

These advantages lead on the one hand to secured transmissions and on the other hand to shorter transmission times in the presence of transmission errors.

3 Network Management

Network management [10] in the automotive area handles the controlled coordinated startup and shutdown of the communication of multiple ECUs within a network. The shutdown of the network (and the accompanying transitions of the ECUs into a low-power or even a power-down mode) is in general done to reduce the network's power consumption in situations, where the functions of the respective ECUs are not required (e.g., when the car is safely stored in a garage).

3.1 Basic Operation

The basic idea of network management (NM) is that the application of each ECU can decide whether or not the ECU requires network communication. In case it does not require network communication, the application requests its local network management instance to transit into sleep mode. This transition however is not performed instantaneously. The network management instead sets the *sleep indication flag* within its periodically transmitted *NM frame* (see Sections 3.2 and 3.3). This way the whole ECU signals its desire to go to sleep mode to all other ECUs. Other ECUs that agree with the decision to transit into sleep mode can signal this agreement by setting the *sleep acknowledge flag* within their own cyclically transmitted NM frames. As soon as all ECUs agree on the decisions to transit into sleep mode, this transition really takes place. As long as a single ECU objects to this decision (since it for example still requires network communication) all ECUs stay awake.

A transition between the awake mode and the sleep mode is indicated to application by means of callback functions. Within this callback function, the application can perform proper actions according to the actual transition (e.g., put the whole ECU into a low power mode in case of a transition to sleep mode).

3.2 Frame Format

The frame format for the FlexRay NM (depicted in Figure 6) is chosen in a way to be compatible to the OSEK NM. Hereby each NM frame consists of a total of 8 bytes.

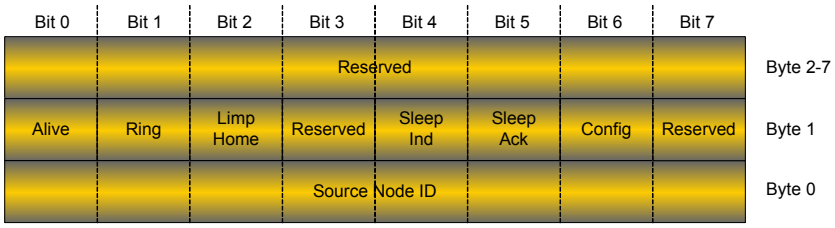

Figure 6: FlexRay Network Management Frame Format

Byte 0 of the frame holds the source node ID for compatibility reasons. Byte 1 contains a set of flags used by the NM protocol. While the sleep indication and the sleep acknowledgement flag are really needed by the protocol, the *ring flag* is just kept for OSEK compatibility and is always set. All other bits within byte 1 and the remaining six bytes (byte 2 to 7) are reserved for future extensions.

Note that neither source node ID nor target node ID have to be transmitted explicitly within NM frame. The source node ID can be determined solely from the point in time the transmission of the NM frame takes place (see Section 3.3). Nevertheless the source node ID is transmitted in order to keep the NM frame format compatible with the OSEK NM frame format. The transmission of the target node ID is not required either, since the FlexRay NM uses logical broadcast communication.

3.3 FlexRay Schedule

TimeCore's network management is built on top of the FlexRay communication system, where the first slot within the dynamic part is used for the exchange of network management data using so-called network management frames amongst the ECUs participating in network management. – This slot is temporally multiplexed between all participating nodes, which results in predictable transmission[1] of the network management data and low bandwidth consumption. The transmission schedule after which the pattern of NM frames is repeated (i.e., each NM participant has transmitted its own NM frame) is called a network management cycle.

In order to achieve the previously described temporal multiplexing, each ECU compares the current cycle counter value of the FlexRay communication system (modulo the length of a network management cycle) with its own network management ID. In case of a match, the ECU is allowed to transmit within the first dynamic slot of this communication cycle. Otherwise some other ECU is allowed to transmit and all other ECUs have to receive this transmission.

3.4 Configuration

Similar to the transport layer the implementation of the network management layer consists of static code sections and configuration files. The configuration files contain the settings for the network management layer like allocated buffers as well as send and receive slots and a unique network management ID for each ECU.
The transport layer set-up is configured for all ECUs using a dedicated configuration tool. This tool is implemented as plug-in of the TimeCore toolchain. Figure 7 illustrates the GUI of the configuration tool. Hereby the GUI elements labeled Network Management provide the possibility to reserved minislots within the dynamic segment of FlexRay for use by the network management algorithm.

[1] In a valid FlexRay configuration (i.e., dynamic part is at least long enough to accommodate the first dynamic slot), transmission within the first dynamic slot is guaranteed to take place.

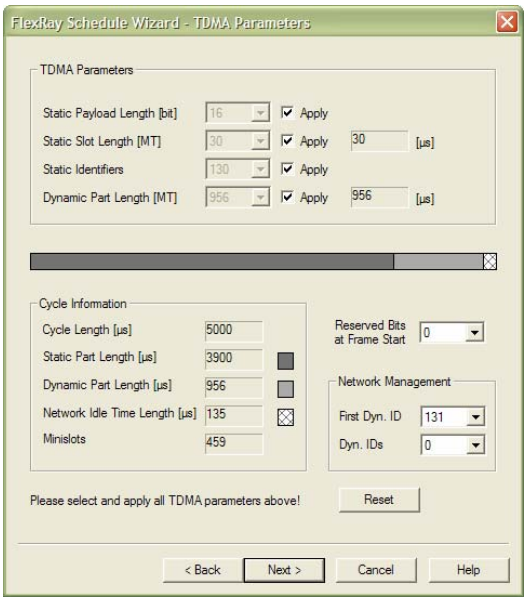

Figure 7: Network Management Configuration

3.5 Comparison to OSEK NM

When comparing the FlexRay NM to OSEK NM, FlexRay NM provides the following advantages:

- **Prevention of Network Congestion Upon Power Up:** Since FlexRay NM uses a static cyclic transmission pattern that is controlled by the global timing of the FlexRay communication system, the network load produced by FlexRay NM is constant in all possible scenarios. OSEK NM however produces high network load when multiple ECUs are powered up simultaneously, since each of the ECUs starts transmitting NM frames in an uncoordinated unsynchronized fashion until the logical ring structure of OSEK NM is established.
- **Fast Response:** Since FlexRay NM makes use of the broadcast nature of the communication, the response time (i.e., the time between the request of going into sleep mode (under the assumption that none of the other ECU objects this decision) and the actual transition into sleep mode) of FlexRay NM is much smaller than the response time of the OSEK NM, where is sleep indication has to be propagated through the whole logical ring.
- **Deterministic Operation in Faulty Scenarios:** A major drawback of OSEK NM is the fact that faulty scenarios lead to indeterminism. The failure of a single ECU causes the logical ring to be broken. The mending of the broken ring requires additional communication which induces additional network load. In FlexRay NM failures of single ECUs are consistently perceived by all recipients and thus no additional communication is required. This results in a deterministic operation even in faulty scenarios.

These advantages lead to a fast, deterministic operation of the FlexRay NM with defined network load even under faulty conditions.

4 Conclusion

TimeCore provides a combination of the time-driven operating system OSEKtime OS, the event-driven operating system OSEK/VDX OS, and the fault-tolerant communication layer OSEKtime FTCom.

With its transport layer, TimeCore provides the basis for higher-level services like diagnostics and flash download – services required and used by almost every automotive company for software updates and diagnostics in the field.

TimeCore's network management facilitates the coordinated startup of all ECUs connected to the network as well as the transition of the ECUs into a low-power or even a power-down mode in order to reduce power consumption in dedicated operation modes of the car (e.g., when placed in the garage).

By providing these two essential services in addition to its predictable time-driven operating system and its fault-tolerant communication layer, TimeCore again underlines that it is well suited for deployment in today's and future automotive applications.

References

[1] *AUTOSAR Standard Software Architecture Partnership Takes Shape*, The Hansen Peport on Automotive Electronics, Vol. 17, No. 8, Portsmouth, NH, USA, October 2004.

[2] Thomas M. Galla and Jochen Olig, *Standard Software Components for X-By-Wire Networks*, Proceedings of the Embedded World 2004 Conference, February 17th-19th 2004, Nürnberg, Germany

[3] V. Barthelmann, A Schedl, E. Dilger, T. Führer, B. Hedentz, J. Ruh, M. Kühlewein, E. Fuchs, Y. Domaratsky, A. Krüger, P.Pelcat M. Glück, S. Poledna, T. Ringler, B. Nash, and T. Curtis, *OSEK/VDX - Time-Triggered Operating System, Version 1.0*, July 24th 2001; Available at http://www.osek-vdx.org/mirror/ttos10.pdf

[4] Thomas Wollstadt, Wolfgang Kremer, Jochem Spohr, Stephan Steinhauer, Thomas Thurner, Karl Joachim Neumann, Helmar Kuder, François Mosnier, Dietrich Schäfer-Siebert, Jürgen Schiemann, Reiner John, Salvatore Parisi, Andree Zahir, Jan Söderberg, Piero Mortara, Bob France, Kenji Suganuma, Stefan Poledna, Gerhard Göser, Georg Weil, Alain Calvy, Karl Westerholz, Jürgen Meyer, Ansgar Maisch, Manfred Geischeder, Klaus Gresser, Adam Jankowiak, Markus Schwab, Erik Svenske, Maxim Tchervinsky, Ken Tindell, Gerhard Göser, Carsten Thierer, Winfried Janz, and Volker Barthelmann, *OSEK/VDX – Operating System, Version 2.2.2*, July 5th 2004; Available at http://www.osek-vdx.org/mirror/os222.pdf

[5] A. Schedl, E. Dilger, T. Führer, B. Hedenetz, J. Ruh, M. Kühlewein, E. Fuchs, T. M. Galla, Y. Domaratsky, A. Krüger, P. Pelcat, M. Taï-Leung, M. Glück, S. Poledna, T. Ringler, B. Nash, and T. Curtis, *OSEK/VDX - Fault-Tolerant Communication, Version 1.0*, July 24th 2001; Avalable at http://www.osek-vdx.org/mirror/ftcom10.pdf

[6] R. Mores, G. Hay, R. Belschner, J. Berwanger, C. Ebner, S. Fluher, E. Fuchs, B. Hedenetz, W. Kuffner, A. Krüger, P. Lohrmann, D. Millinger, M. Peller, J. Ruh, A. Schedl, and M. Sprachmann, *FlexRay - The Communication System for Advanced Automotive Control Systems*, SAE 2001 World Congress, Detroit, MI, USA, March 2001

[7] T. Führer, F. Hartwich, R. Hugel, H. Weiler, *FlexRay - The Communication System for Future Control Systems in Vehicles*, SAE 2003 World Congress & Exhibition, Detroit, MI, USA, March 2003

[8] ISO (International Organization for Standardization), *Road Vehicles – Diagnostics on Controller Area Networks (CAN) – Part 2: Network Layer Services*, ISO/DIS 15765-2.2, 1, rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland, June 2003

[9] ISO (International Organization for Standardization), *Road Vehicles – Diagnostics on Controller Area Networks (CAN) – Part 3: Implementation of Diagnostic Services*, ISO/DIS 15765-3.2, 1, rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland, October 9th 2003

[10] C. Hoffmann, J. Minuth, J. Krammer, J. Graf, K. J. Neumann, F. Kaag, A. Maisch, W. Roche, O. Quelenis, E. Farges, P. Aberl, D. John, L. Mathieu, M. Schütze, D. Gronemann, and J. Spohr, *OSEK/VDX – Network Management - Concept and Application Programming Interface, Version 2.5.2*, January 16th 2003; Available at http://www.osek-vdx.org/mirror/nm252.pdf