Johannes Gosda

AUTOSAR

Communication Stack

March 31, 2009

Summary. The drastically increasing of complexity within the automotive domain due to an increasin number of embedded computing units per car leads to the need of a framework for accomplishing this high complexity. Therefore AUTOSAR was developed. The AUTOSAR framework provides specifications for communication modules, which are described in this document. The focus lies on the AUTOSAR Communication Stack and its Basic Software Modules associated with the communication. The different kinds of signals traversing the communication stack up and down are introduced as well as the different softwarelayers they belong to, like Communication Services Layer, Communication Hardware Abstaction Layer and Communication Drivers Layer.

Contents

1

Introduction

The automotive electronics market is exposed to an rising demand from customers in the domains of comfort, safety and fuel efficiency. The basic functionality within a car is solved by hardware, while software handles the multiple extra functions. Because of hardware devices alone can not satisfy the consumers needs and wishes, the Original Equipment Manufacturers (OEMs) are exploring new markets with new software applications and features. Therefore a competition in mounting software on the available hardware starded and led to a higher complexity of solutions. Because of the automotive electronics has evolved from component level solutions to system level architecture, testing and maintenance became more and more difficult. [1].
The drastically increasing of complexity within cars due to an increasing number of embedded computing units (ECUs) per car [2] leads the automotive industry to develop methods to cope with this complexity. OEMs are looking for solutions at an application level while semiconductor suppliers started to work together with software developers because of high price pressures and the possibility of providing complete solutions to OEMs.

1.1 Scope of this Document

This document shall not explain the autosar concepts in detail. The knowledge about the concepts and the rough structure of the AUTOSAR architecture is assumed to be present and only a short introduction to the AUTOSAR concept is given in the beginning of this paper after introducing the motivation for the AUTOSAR framework. Then the differences between signals, signal groups and PDUs, which are all necessary for communication, are explained. After this, the focus will be on the Basic Software Communication Modules. The AUTOSAR COM, PDU Router and IPDU Multiplexer are described more detailed than the other BSW modules. The Network Management is only short introduced. At the end there is a more detailed description of the transmission of signals and an explanation of the determinition of the

transmission modes of PDUs. Complex device drivers are not scope of this document, even if they are associated with communication. Within this paper parts of the Basic Software other than Communication Services, Communication Hardware Abstraction and Communication Drivers are neglected.

1.2 AUTOSAR

The partnership of automotive manufacturers and suppliers develop and establish together the AUTomotive Open System ARchitecture (AUTOSAR). AUTOSAR is a de-facto open industry standard ror automative E/E architectures. The automotive engineering environments change mentioned in chapter 1 makes the members of the automotive industry developing solutions to manage the increasing E/E complexity associated with the growth in functional scope, to improve flexibility, scalability, quality and reliability of their systems and to detect errors in early design phases [3].
AUTOSAR shall provide an opportunity for seamless tool chain, an enhancement of software quality, the possibility of developers to concentrate on functions with competitive value. More detailed benefits of the AUTOSAR framework can be found in [3] and [4].

1.2.1 AUTOSAR Concept

The basic approach of AUTOSAR is providing a framework for developing application components independet from the hardware. Applications shall be implemented in form of AUTOSAR Software Components (SWC) which encapsulate this application running on the AUTOSAR infrastructure. A SWC has well-defined interfaces, which are described and standardized within AUTOSAR. These components are connected via the Virtual Functional Bus (VFB), who provides all communication mechanisms on an abstract and technology independet level. With the VFB AUTOSAR SWCs will be virtual integrated when the connections between each other are defined. This component view alleviates the development of software application, because developers need not to know the specific hardware. This concept will is described in more detail in [5].
From the hardware view there still exist the different ECUs wich are interconnected with different bus systems. Each ECU may have a different hardware-layout like different processors or communication bus systems and additionally different SWCs running on. Therfor AUTOSAR supplies different abstraction layers to provide a common upper interface. Figure 1.1 envisibles the basic AUTOSAR approach.
The AUTSOSAR SWCs interconnected via the VFB have to be mapped to the different ECUs. Because of the fact that AUTOSAR SWCs are programmed against the VFB and are using specified interfaces [6] these mapping can be conducted without changing the SWCs. The Runtime Environment (RTE) is

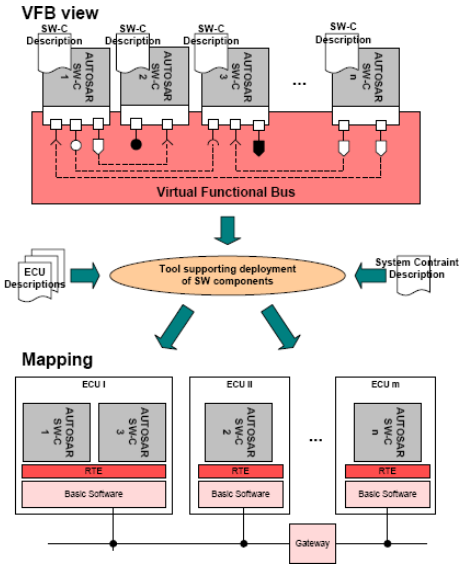

Fig. 1.1. Basic AUTOSAR approch with component view (up) and ECU software architechture and hardware topology (down) [3]

the, with tool support generated, implementation of the VFB from the compenent's view. AUTOSAR offers a methodology described in [7] for creating the resulting RTE and further software running on an ECU.

1.2.2 ECU Software Architecture

AUTOSAR devides the software situated on an ECU into different layers which are shown in figure 1.2. The most top layer is the application layer that contains application software components, actuator software components and sensor software components. These components communicate via AUTOSAR interfaces [6] and the AUTOSAR RTE with themselfs. Under the RTE and above the ECU hardware is located the AUTOSAR Basic Software. The Basic Software is the standardized software providing necessary services for running the functional part of the software.
The Basic Software (BSW) conains services for e.g. diagnostic protocols and memory managemant. Communication, I/O management and network management are part of the BSW as well as an Operating System. The BSW can be divided monolithically into the Services Layer, Hardware Abstraction Layer and Microcontroller Abstraction Layer (MCAL). The ECU abstraction decouples higher-level software from all underlying hardware dependencies by providing a software interface to the electrical values of any specific ECU. The MCAL ensures a standard interface to the BSW modules by managing the microcontroller peripherals and by providing microcontroller independent

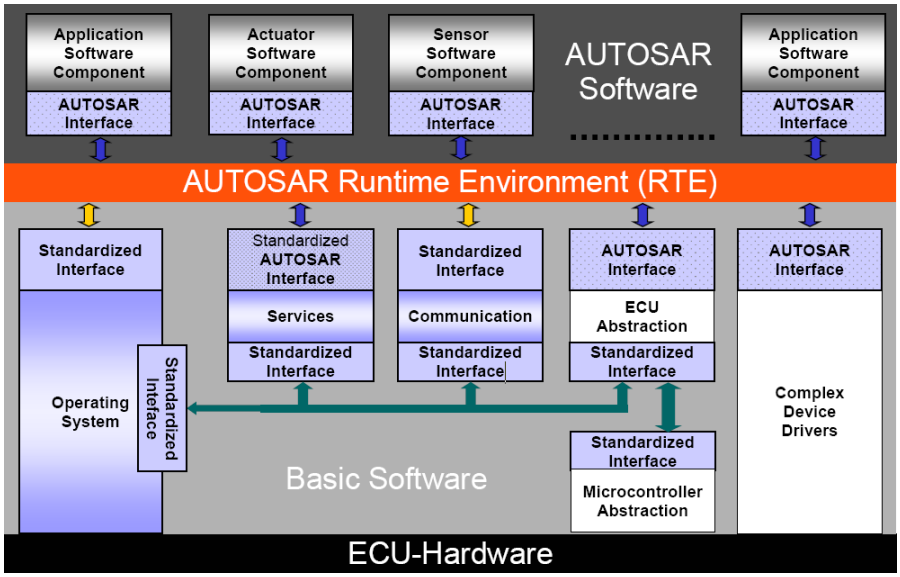

Fig. 1.2. Overview of Software Layers [8]

values. As visible in figure 1.3 these layers themselfs can be devided by their purpose.

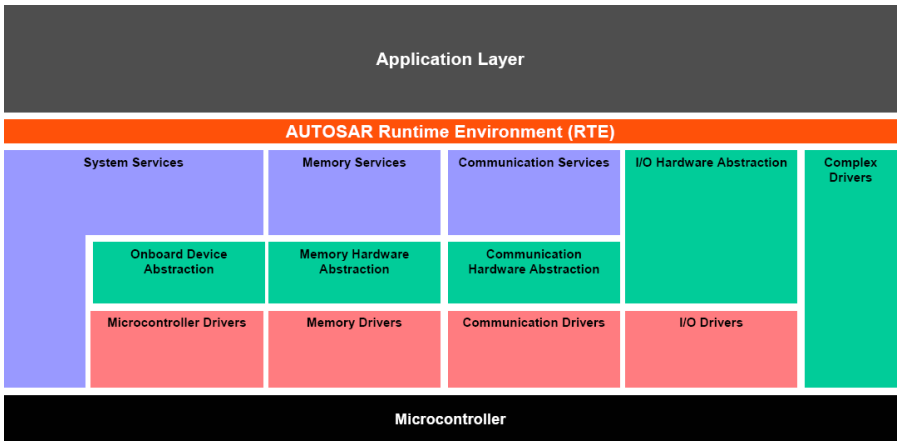

Fig. 1.3. Overview of Software Layers in detail [8]

1.3 Implementation and Simulation

Different tools already support modelling AUTOSAR systems and RTE code generation. Sebastian Wätzoldt shows in his paper [9] how AUTOSAR systems can be modeled within dSPACE SystemDesk. He displays the modelling of components including integrating runnables, integrating the hardware topology of the system, the mapping of the network communication and the RTE and Operation System (OS) generation.
The paper [10] deals with the simulation of automotive systems within the different development process stages and points which kind of simulation is possible at that stages. There is also described which tools can be used to simulate AUTOSAR architectures and a short indroduction to the use of dSPACE SystemDesk and TargetLink for simulating these architectures is given.

2

Signal, PDU and SDU

2.1 Signal

An AUTOSAR signal is similar to a signal in AUTOSAR COM but not always the same (the transformation may change the syntax of the signal for complex data tpyes) and an AUTOSAR COM signal is the same as a message in OSEK COM [11]. The RTE performs the transformation from an AUTOSAR Signal to a signal in COM and an AUTOSAR signal is carried by one or more signals in COM [12].

2.2 Signal Group

In AUTOSAR there can be used so called complex data types. Inside a complex data type there are one or more data elements and to ensure data consistency a complex data type must be treated as an atomic unit [13]. A set of signals that must always be kept together in a common I-PDU is presented by a signal group. With a signal group can be guaranteed that AUTOSAR composite data types are transfered atomicly. Every signal can belong to at most one signal group and a signal group can belong to at most one I-PDU but signal groups could contain no signals (then they are empty). The signals that belong to the signal group are contiguous. Which signals are grouped to which signal group is assumed to be an input for the COM generation process [12].
Update-bits supported by AUTOSAR COM are a mechanism which provides for the receiver the identification whether the sender has updated the data in this signal or signal group before sending. These Update-bits are not allowed if direct/n-times transmission mode with n greater 1 is used.

2.3 PDUs

2.3.1 PDU

PDU is the abbreviation of Protocol Data Unit. It contains SDU and PCI. Each PDU has a static PDU ID which is used to identify PDUs [14]. On PDU transmission the upper layer sends its PDU to the lower layer, which interprets this PDU as the SDU of its own PDU. This mapping depicts figure 2.1. Non-TP I-PDUs shall not exceed a length of 8 bytes. This ensures that an I-PDU can be transmitted in a single CAN message [15].

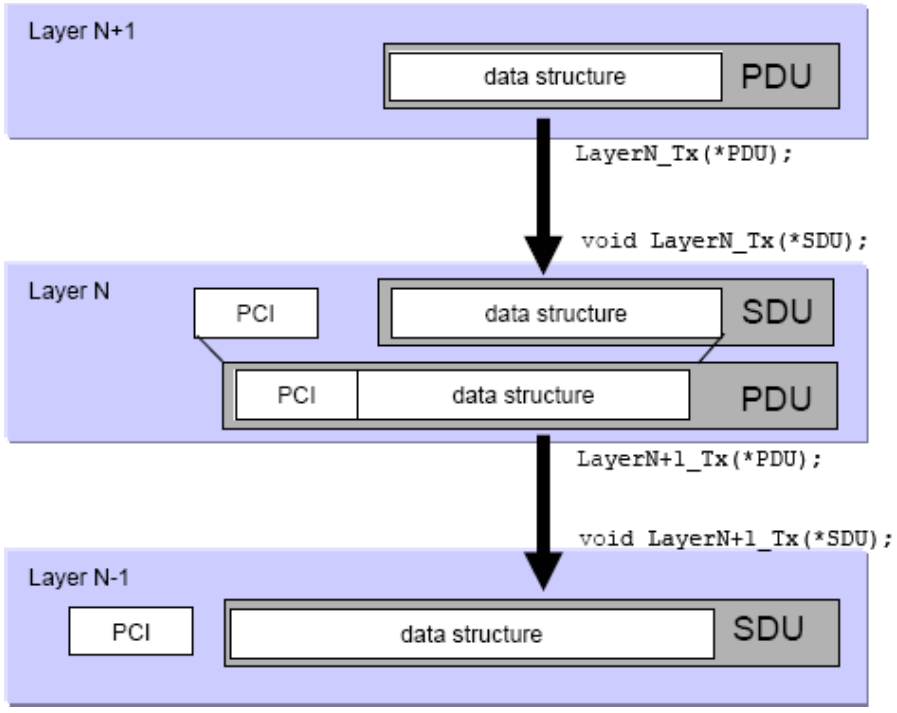

Fig. 2.1. Packing of SDU into PDU over different layers [8]

2.3.2 SDU

SDU is the abbreviation of Service Data Unit and is a part of a PDU. The SDU is the data passed by an upper layer with the request to transmit the data. It is as well the data which is extracted after recaption by the lower layer and passed to the upper layer [8].

2.3.3 PCI

Protocol Control Information (PCI) is the information that is needed to pass a SDU from one instance of a specific protocol layer to anonther instance. On transmission side there is added the PCI by the protocol layer and removed on receiving side. In a PCI is contained e.g. source and target information [8].

2.3.4 DLC

DLC means Data Length Code and is part of an L-PDU. The DLC describes the length of the related SDU [16].

2.3.5 PDU Naming Conventions

PDUs can hold a layer specific prefix to distinguish the PDUs of the different layers from each other. The PDUs are devided into I-PDUs, L-PDUs and N-PDUs.
An I-PDU is used for the data exchange of the modules directly above the PDU Router. These mudules are ATUOSAR COM and AUTOSAR DCM. The Interaction Layer Protocol Data Unit (I-PDU) is assembled and disassebled in AUTOSAR COM. It consists of one or more signals [12]. Every I-PDU must belong to exactly one I-PDU group.
The maximum length of an I-PPDU depends of the maximum length of the L-PDU of the underlying communication interface because the I-PDUs of COM are passed via the PDU router directly to the communication interfaces. An L-PDU is a Data Link Layer Protocol Data Unit which is assembled and disassembled in AUTOSAR Hardware Abstraction layer. In AUTOSAR the Data Link Layer is equivalent to the Communication Hardware Abstraction and Microcontroller Abstraction Layer [13]. The L-PDU consists of Identifier, DLC and data (L-SDU). The maximum length of a CAN and LIN L-PDU is 8 bytes and for FlexRay the maximum length of the L-PDU is 254 bytes.
An N-PDU is a PDU of the network layer and this denotation is used in the AUTOSAR TP Layer.

2.3.6 I-PDU Group

An I-PDU group is a collection of I-PDUs in COM and contains zero or more I-PDUs or I-PDU groups. But in can only either hold I-PDUs or I-PDU groups. An I-PDU that is part of another I-PDU group can not contain another I-PDU group so the I-PDU group hirarchy is limited to two (figure 2.2). No I-PDU group can be included in more than one other I-PDU group and a I-PDU group must not contain itself. A mixture of received I-PDUs and sent I-PDUs in a single I-PDU group is not allowed. [12]
The number of I-PDU groups is limited to 32. I-PDU groups are stopped by default. AUTOSAR COM provides routines for starting and stopping I-PDUs.

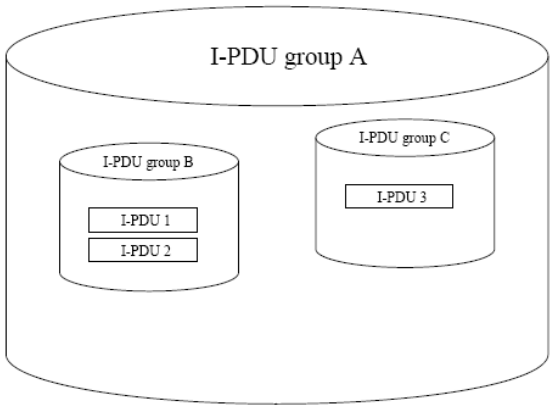

Fig. 2.2. Grouping of I-PDUs and I-PDU groups [13]

Stopping I-PDU groups shall disable the transmission of I-PDUs which belong to the I-PDU group and only the Com_TriggerTransmit function is processed because this can not be prohibited by AUTOSAR COM. In an I-PDU group containing I-PDU groups is started or stopped, the contained groups shall also be started or stopped.

2.4 I-PDU Multiplexing

I-PDU multiplexing is performed by the IPduM (ch. 3.2.3) and means using the same I-PDU ID transferred from the PDU-Router to the Communication Hardware Abstraction Layer with more than one unique layout of this I-PDU [17, p. 19f]. The IPduM always comines only two I-PDUs of COM to a multiplexed I-PDU because in COM there is one COM I-PDU for the static part and one COM I-PDU for each layout of the dynamic part of one IPduM I-PDU. Within COM the static and the dynamic parts are treated as separated I-PDUs with their own I-PDU IDs. Figure 2.3 shows the possible layout of a multiplexed I-PDU.

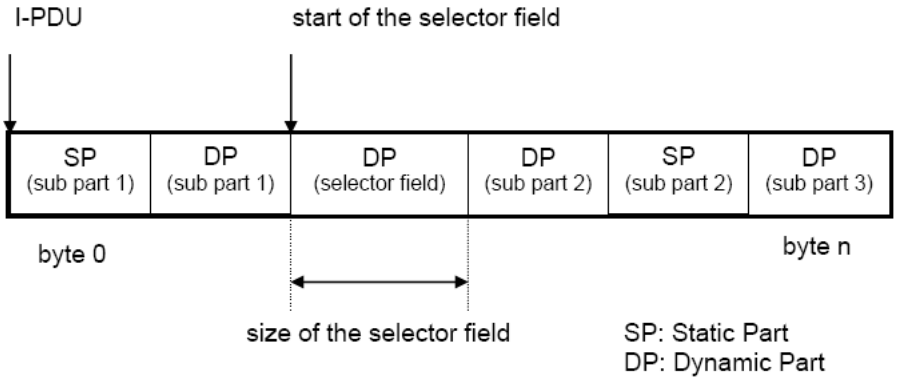

Fig. 2.3. Possible layout of a multiplexed I-PDU [17]

3

Basic Software Communication Modules

3.1 Communication Stack Modules

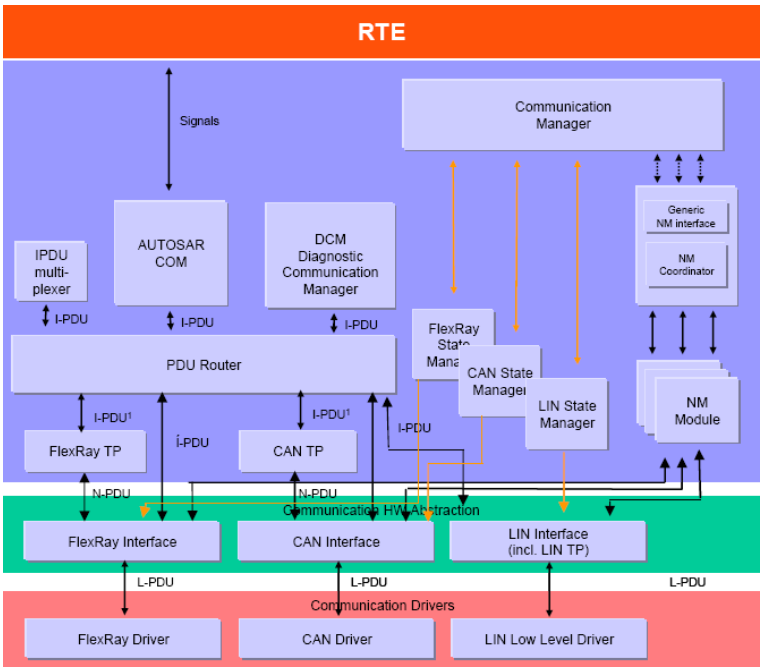

Fig. 3.1. Communication Stack Modules and structure [14]

3.2 Communication Services

3.2.1 AUTOSAR COM

A main-feature of AUTOSAR COM is the provision of a signal oriented data interface for the RTE and its single lower layer interfaces is the PDU Router. COM controls the communication transmission which means e.g. starting and stopping of I-PDU groups. It is sending signals congruent to the signals transmission type (see also: ch. 4.2) specified in the VFB specification. Minimum distances between transmission requests involving COM shall be guaranteed. There is a filtering mechansims and different notification mechanisms for incoming signals. COM supports monitoring of receive signals to notice possible signal timeouts. Initial values and update indications are provided as well as endianness conversion or sign extension if necessary. AUTOSAR signals are packed and unpacked by COM to I-PDUs to be transmitted and the received signals are provided to the RTE. The signals or signal groups are routed from received I-PDUs into I-PDUs to become transmitted by the AUTOSAR COM module.
Gateway functionality on signal level aka signal-based gateway is provided by the Signal Gateway, which is integral part of COM. The signal gateway provides forwarding signals and signal groups in a 1:n manner and the signal gateway provides mapping of signals or groups of signals (Complex Data Types) initiated by a signal routing trigger. Signal routing trigger is generated by COM core funtionality. Signal gateway uses packing/unpacking mechanisms and timeout handling mechanisms of COM. If no signal routing functionality is needed, the gateway should scale down to no size. The destination of a signal or of a signal group is determined by using unique static names of the signals and signal groups and by using a configuration table [13].
COM shall provide support for endiannes conversion of all integer types to support the types boolean, uint8, uint16, uint32, sint8, sint16, sint32, uint8[n]. The conversion shall take place before notification detection on receiver side [13].
If an I-PDU is received and shall be unpacked there are two different configurable signal indication modes [13]:

- Immediate: the signal indications / confirmations are performed in "Com_RxIndication"
- Deferred: singal indication / confirmations are deferred for example to a cyclic task

3.2.2 PDU Router

The PDU Router module must be instantiated in every AUTOSAR ECU. Its main task is routing I-PDUs between the Communication Services and Hardwareabstraction Layer modules. The parts PDU Router routing tables and PDU Router Engine compose the PDU Router. The static routing tables

can be updated post-build time in the programming state of the ECU and are describing the routing attributes for each PDU that shall be routed. The PDU Router Engine is the actual code thats performs the routing while embracing the routing tables. The supported communication is shown in figure 3.2. In addition to the routing algorithm the PDU Router Enginge provides a minimum routing capability for routing specific PDUs without using the PDU Router routing tables. This supports accessing the DCM for activation of the ECU bootloader even when post-build time configurable PDU Router routing tables are corrupted. The set of PDUs for the minimum routing capability can not be changed after build-time [14].
There are three different classes of operations the PDU Router modules performs, which are PDU reception receive I-PDU and send to upper layer, PDU transmission (transmit I-PDU on request of an upper layer) and PDU gateway (receive and I-PDU from an lower layer and immediately send it via the same or another lower layer module).
The PDU Router provides Gateway functionality on PDU level aka frame-based Gateway. A special case is routing on-the-fly, which means that the transfer of TP data is started although not the full TP data is yet buffered. Therefore the gateway provides to the receifing TP module a smaller buffer than the overall size of the data and when it is filled, the gateway starts transmitting these data on the destination bus. The receiving TP module gets another buffer in parallel to continue receiving the data.
In addition the PDU Router provides routing of PDUs between COM and CAN/LIN/FlexRay interfaces and between DCM (see: 3.2.8) and TP (see: 3.2.4, 3.3.3) modules, and between the communication interface layers and between TP modules [15]. Figure 3.2 depicts the different communication traces including the communication between PDU Router and IPduM.
The routing operation of the PDU Router module does not modify the I-PDU itself. The I-PDU is simply forwarded to the destination module. To allow the drectly upper modules DCM and COM and the lower level TP modules and communication interfaces to communicate with the PDU Router, it provides an API for all these modules [14]. In addition an interface for the IPduM is provided. The PDU Router supplies the Indication of incoming multiplexed I-PDUs, a sending interface for outgoing I-PDUs and a confirmation of I-PDUs which went out to the IPduM [17]. All these mentioned interfaces are constructed in that way, that the operations required to pass the data beween the upper and lower layers are minimized.
The PDU Router is initiated by a PDU routing trigger which may be generated by the CAN, LIN, or FlexRay interfaces, the corresponding TP modules, the service layers COM and DCM or IPduM. A multicast operartion is the transmission of PDUs to a group of receivers [14].
The Routing Configuration comprehends configuration data that controls the operation of the PDU Router and Signal Gateway and should be encapsulated to allow updates. With the configuration data is defined which destination each PDU of the PDU Router and each Signal of the signal gateway have

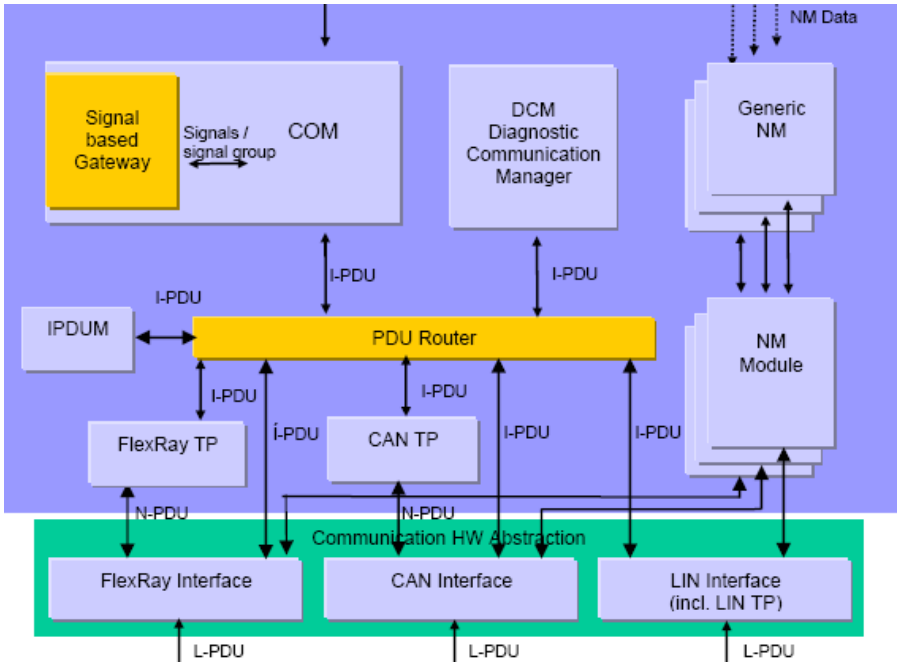

Fig. 3.2. Different possibilities for communication paths relating the PDU Router

[15]. The PDU Router determines the destination of a PDU by using this static configuration table and the static PDU ID.
The routing layer is statically configurabele per ECU and its size is ECU specific. If no gateway functionality is needed, the routing layer can be configurated down to zero size.
The PDU Router is not responsible for network management data exchange because this communication is separated from the PDU Router. It also is not in authority for signal extraction or conversion mechanisms, data integrity checking, modifying I-PDUs, PDU payload dependent routing decisions, routing between TP modules and communication interface modules.

3.2.3 IPDU Multiplexer (IPduM)

The AUTOSAR IPduM is situated within the Communication Services part of AUTOSAR Basic Software next to the PDU Router (ch. 3.2.2) as can be seen in figure 3.3. The IPduM is used for multiplexing of I-PDUs which is currently known from CAN [18].
An multiplexed I-PDU is an I-PDU with more than one unique layout of its SDU but therfor using the same PCI (see ch. 2.3 for details). The SDU is containing a selector field for distinguishisng the different layouts of the multiplexed I-PDUS from each other. The selector field is a set of following bits

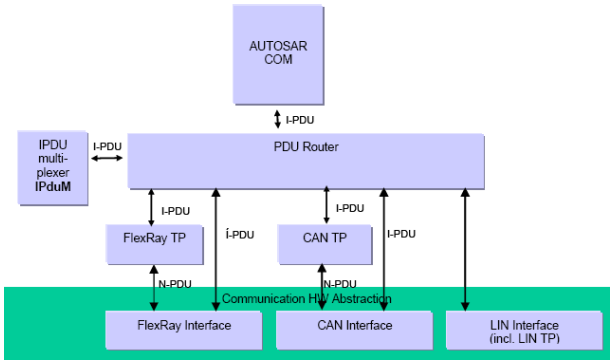

Fig. 3.3. IPduM and its surrounding modules [18]

and determine the layout of the multiplexed part of the I-PDU.
On sender side the I-PDU Multiplexer module receives appropriate I-PDUs from COM (ch. 3.2.1), assembles them to new multiplexed I-PDUs and sends the new I-PDU back to the PDU Router. Receiving these I-PDUs from COM always implies the participation of the PDU Router because this is the sole communication module interacting with the IPduM. On receiver side the IPduM interprets the content of multiplexed I-PDUs and sends the resulting separated I-PDUs back to COM while taking into account the value of the selector field.
The IPduM provides an interface for indication of incoming de-multiplexed I-PDUs to the PDU Router, a sending interface for I-PDUs that have to be multiplexed and a confirmation interface for transmitted I-PDUs [17].

3.2.4 Transport Layer (TP)

For each communication bus system there is a Transport Layer module within the services layer except for LIN. The LIN TP is situated within the LIN IF. It will only be dealt with the CAN TP because describing FlexRay TP and LIN TP aswell whould exceed the scope of this document. These modules are specified in [19] and [20].
The CanTp is needed to segment and reassemble CAN I-PDUs with a lenght greater than 8 bytes. Its main tasks are controlling the data flow, detection of errors in segmentation sessions, transmit cancellation, segmentation of data in transmit direction and reassembling of data in receive direction. The CanTp has an interface to the lower level CanIf and uses a single upper layer module, the PDU Router. The CanTp module is able to deal with multiple connections, with its maximum number specified, simultaneously [21]. Figure 3.4 shows the interaction of the CanTP and its upper and lower layer.

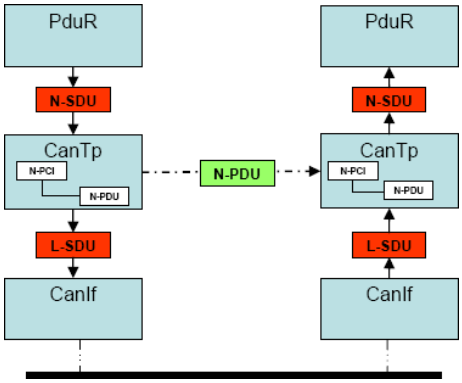

Fig. 3.4. CAN Transport Layer interactions [21]

3.2.5 Communication Manager (ComM)

The ComM is a resource manager which encapsulates the control of the underlying communication services. It controls the basic software modules related to communication and coordinates the bus communication access requests. The ComM shall simplify the usage of the bus communication stack for the user. It shall offer an API for disabling the sending of signals, shall be able to controll more than one communication bus channel of an ECU and shall simplify the resource management by allocating all resources necessary for the requested communication mode [22]. The COM Manager (ComM) controls the starting and stopping of sending and receiving I-PDUs via AUTOSAR COM. The NM is used by the ComM to synchronize the control of communication capabilities across the network.

3.2.6 CAN/FlexRay/LIN Bus State Manager

The actual bus states are controlled by the corresponding Bus State Manager. The actural states of the bus corresponds to a communication mode of the ComM. The ComM requests a specific communication mode from the state manager and the state manager shall map the communication mode to a bus state [22].
E.g. the comM uses the API of the CanSM to request communication modes of CAN neworks. The CanSM uses the API of COM to controll CAN related PDU groups and it communicates with the CanIf to conrol the operating modes of the CAN controllers and to get notified by the CanIf about peripheral events [23].

3.2.7 Network Management Modules (NM)

The Generic Nework Management Interface adapts the ComM to the bus specific network management modules. It provides an inteface to the ComM

and uses services of the network management modules. The bus specific network management modules are CAN NM, FlexRay NM and LIN NM. The AUTOSAR NM Interface can only be applied to communication systems that support broadcast communication and bus-sleep mode [24]. For network management data exchange the PDU Router module is bypassed.

3.2.8 Diagnostic Communication Manager (DCM)

The main purpose of the DCM is providing a common API for diagnostic services. It is used while development, manufactoring or service by external diagnostic tools [25]. In figure 3.5 there is an overview of the communication over the DCM. The DCM performs the scheduling of diagnostic PDUs. It acts as a user by requesting full communication from the ComM if diagnostic shall be performed [22].

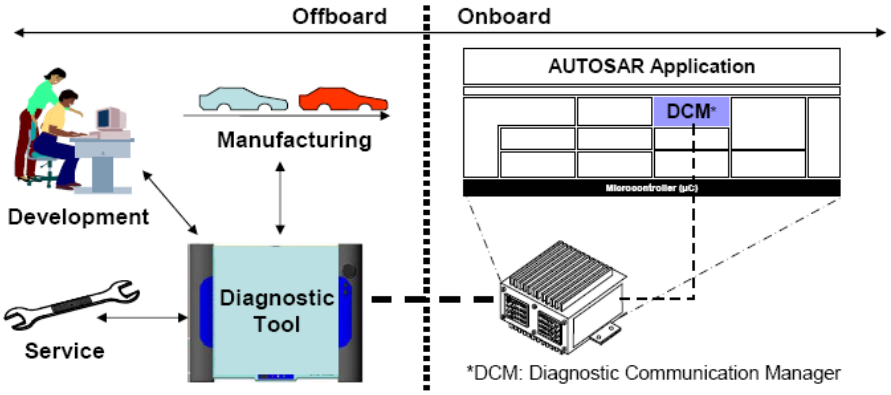

Fig. 3.5. Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application [25]

3.3 Communication Hardware Abstraction

3.3.1 CAN Interface (CanIf)

The CanIF is situared in the communication stack between the lower level CAN Driver and the upper level communication service modules and it represents the interface to the services of the CAN Driver for the upper communication layers, which includes managing the different CAN hardware devices. The CanIf initializes the CAN Driver module during startup phase. It has to call the CAN Driver modules periodical processing function periodically via the interface provided by the CAN Driver module. The CanIf has to check

the validity of state changes for the CAN Driver module, because the CAN Driver does not check for validity of these changes.

3.3.2 FlexRay Interface

The FlexRay interfaces main task is to provide to upper layers an abstract interface to the FlexRay Communication System. The FlexRay interface accesses the hardware not directly but over the hardware specific FlexRay interfaces. More information and detailed specification a depicted in [26] and [27].

3.3.3 LIN Interface

The LIN Interface is designed to be hardware independent and provides transmit requests from the upper layers and receive notifications for the upper layers. For details please refer to [28] and [19].

3.4 Communication Drivers

3.4.1 CAN Driver

The CAN Driver is part of the lowes layer and offers the CAN Interface uniform interfaces to use. It hides hardware specific properties of the CAN Controller as far as possible [29]. The CAN Driver performs the hardware access and provides a hardware independent API to the upper layer, the CAN Interface (CanIf). Services for initiating transmission are offered by the CAN Driver and it calls the callback funtions of the CanIf module for notifying events hardware independently. In addition there are services provided by the CAN Driver module to control the state of all CAN controller belonging to the same CAN hardware unit. A CAN controller serves exactly one physical channel. A detailed description of the CAN bus is given in [30]. A CAN hardware unit is represented by one CAN Driver and either on chip or an external device. It may consist of one or multiple CAN controllers of the same type and one or multiple CAN RAM areas [29]. A single CAN Driver module can handle multiple CAN controllers if they belong to the same hardware unit as shown in figure 3.6.

If an L-PDU shall be transmitted, the CAN Driver writes the L-PDU in a buffer inside the CAN controller hardware and if an L-PDU is received, the CAN Driver module calls the RX indication callback funtion with the L-PDUs ID, the DLC (see: ch. 2.3) and with a pointer to the L-SDU. The CAN Driver can access hardware resources and converts the given information for transmission into a hardware specific format and triggers the transmission. The CAN Driver module offers the CanIf services to control the state of the CAN

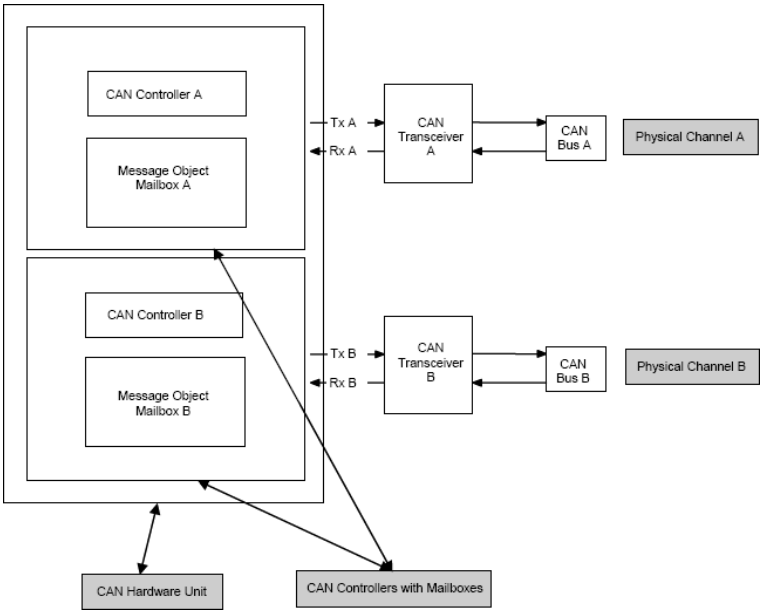

Fig. 3.6. A CAN hardware unit consisting of two CAN controllers connected to one physical channel each. [16]

Controllers by callback fucktions for bus-off and wake-up events. It implements the interrupt service routines for all CAN hardware unit interrupts that are needed. While startup the CAN Driver initializes static variables including flags, sets common settings for the complete CAN hardware unit and sets CAN controller specific settings for each CAN controller.

3.4.2 FlexRay Driver

The use of the FlexRay Driver is abstracting from the hardware related implementation details of spedific FlexRay Communication Controllers. A description of the FlexRay Driver can be found in [31].

3.4.3 LIN Driver

The LIN driver is part of the Microcontroller abstraction layer and performs the hardware access and offers a hardware independent API to the upper layer. For further information please consult [32].

4

Communication

4.1 Application Layer and RTE

Applications written in the context of AUTOSAR consist of components. These components communicate with each other via ports (component view see: 1.1, [3] and [5]). The communication between two components can consist of a single (AUTOSAR) signal or a whole signal group. From the view of the AUTOSAR SWC it is not known at implementation time, which communication media is used. All bus specific replications of send requests by a SWC to underlying layers and bus specific timing behavior must be done by COM or by the appropriate bus interfaces and drivers [13]
The RTE uses the capability to send and receive signals of AUTOSAR COM. Figure 4.1 shows the VFB's send modes corresponding to the transfer property of a signal and the transmission mode of an I-PDU described in ch. 4.2.

Transfer property (horizontal) ***Transmission mode (vertical)***	***Triggered***	***Pending***
Direct/N-Times	N-Times	--
Periodic	--	Cyclic
Mixed	N-Times	Cyclic
None	--	--

Fig. 4.1. Mapping of transfer property and transmission mode to send modes of the VFB [13]

After a send request from an upper layer for a specific signal, the signal is written to the appropirate I-PDU buffer as defined by configuration and the selection of the transmission mode of the I-PDUs in done.

4.2 Transmission Modes and Transmission Mode Selection

COM shall provide different transmission modes for each I-PDU [12].

- Periodic: transmissions occur indefinitely with a fixed period between them
- Direct / n-times: event driven transmission with n-1 repetitions
- Mixed: periodic transmission with direct/n-times transmissions in between
- None: no transmission

Two of these transmission modes shall be supported for each PDU so that it will be possible to switch between both modes during runtime. To decide which of the two transmission modes is selected, COM shall provide the possibility to attach a condition to each signal within an I-PDU separately. If all conditions that are defined for signals within one specific I-PDU evaluate to true then one transmission mode shall be used for this I-PDU. If at least a single condition defined for a signal within this I-PDU evaluates not to true, then the other mode shall be used. These conditions shall be checked directly if a related signal or signal group is sent by the RTE.
The attached condition on a signal for evaluating the transmission mode for an I-PDU is called transfer property. A transfer property of a signal can either be triggered or pending. A transfer property of a signal with the triggered-value causes an immediate transmission of the I-PDU except if transmission mode periodic or none is defined for the I-PDU. If the transfer property of a signal is pending, no transmission of an I-PDU is caused.
Because of I-PDUs can contain more than one single signal there is needed a method to derive the I-PDU's transmission mode from the state of signals that are contained in one specific I-PDU. For this method signals within a signal group are treated like normal signals.
For each I-PDU there is defined a Transmission Mode Selector (TMS). The TMS is calculated by evaluating the Transmission Mode Conditions (TMC) of a configurable subset of signals belonging to the specific I-PDU. The TMS is defined to be true if at least one TMC of the configurable subset of signals evaluates to true. If all TMCs evaluate to false the TMS is defined to be false. If Com_SendSignal or Com_SendSignalGroup are called, the TMS of the I-PDU shall be re-calculated. Figure 4.2 shows the mapping of signals into an I-PDU and the evaluation of the TMS. A detailed description of the selection of transmission modes is situated in [13, ch. 7.4.3.3].

4.3 Communication Interaction

Here is a short example for the path of interaction for sending a signal. An application is deciding internally to update e.g. an actuators value. Here is expected, that the application already requested its communication mode from the ComM. The application will only call a Write_Signal_... function.

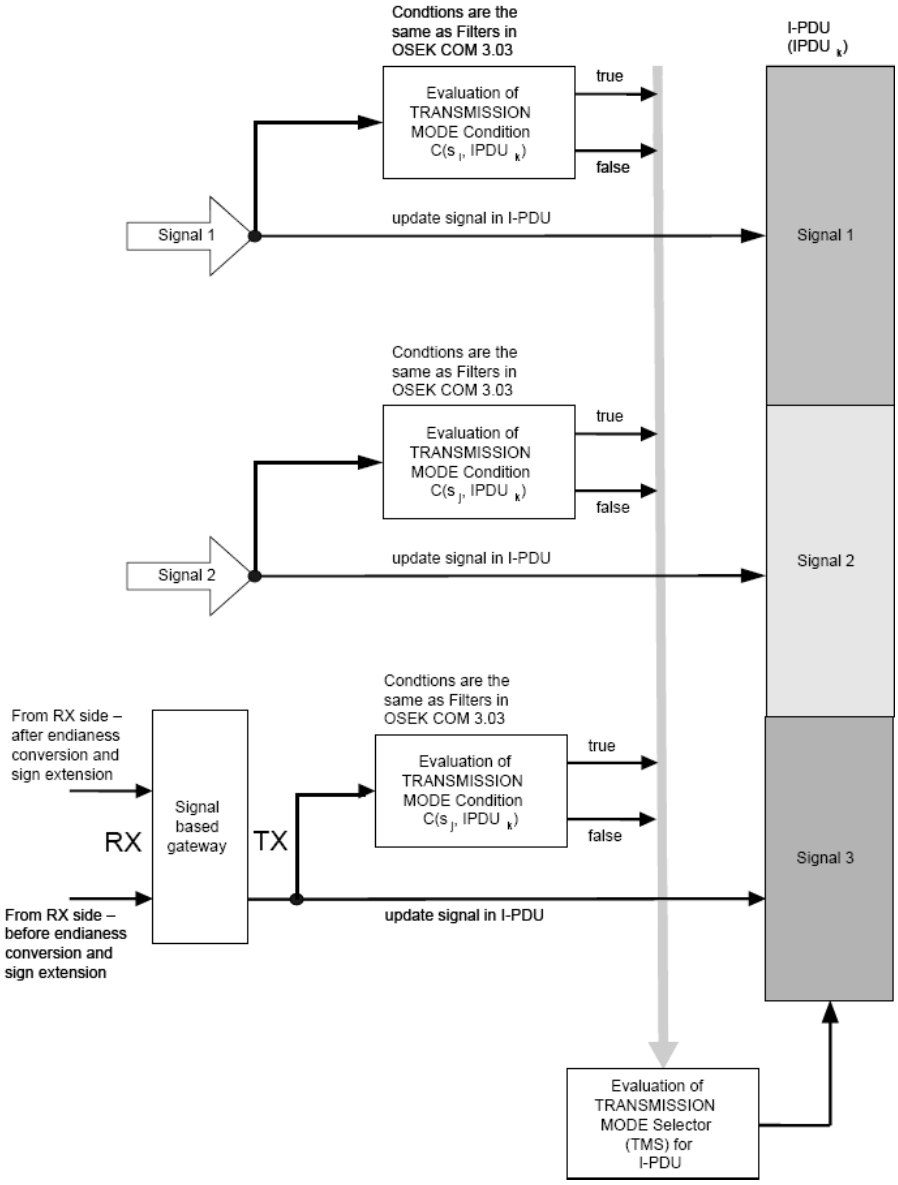

Fig. 4.2. Signal flow and TMS evaluation [13]

This function will be, while generating the RTE, mapped to a RTE_Write_... function. Within this funtion there will be either a Com_SendSignal or a Com_SendSignalGroup (or Com_UpdateShadowSignal) function call, depending on whether the signal to send was a normal AUTOSAR signal or an AUTOSAR complex data type. In COM the signal is packed into a PDU and the signals transferproperty is evaluated to determine the I-PDUs transmission mode. The AUTOSAR COM module calls the PduR_ComTransmit function for forwarding the PDU to the PDU Router. The PDU Router gets based on the PDU ID and its PDU Router routing table the target of the PDU and sends it e.g. to the CanIf with the use of CanIf_Transmit. The CanIf calls Can_Write from the CAN Driver that will copy the L-PDU into the CAN hardware. If the PDU was sent on the bus the CAN hardware triggers a transmit interrupt at the CAN Driver. The CAN Driver then calls CanIf_TxConfirmation, the CanIf calls PduR_TxConfirmation and the PduR calls Com_TxConfirmation. Sequence diagramms of the transmission, confirmation and receive indication operation can be found in ch. 9 in [16, ch. 9], [33, ch. 9.1, 9.3], [14, ch. 9] and [13, ch. 9].

References

1. Rajagopalan, R.: Automotive software market - mix of opportunities and implications. Frost & Sullivan (September 2004)
2. Murphy, M.: The rise and fall of ecu numbers per vehicle. Automotive World (October 2007)
3. AUTOSAR GbR: Technical Overview. 2.2.2 edn. (August 2008)
4. Schlegel, J.: Technical foundations for the development of automotive embedded systems. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
5. Warschofsky, R.: Autosar software architecture. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
6. Naumann, N.: Autosar runtime environment & virtual function bus. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
7. Hebig, R.: Autosar methodology & templates. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
8. AUTOSAR GbR: Layered Software Architecture. 2.2.2 edn.
9. Wätzoldt, S.: Modeling and development of autosar using systemdesk. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
10. Krasnogolowy, A.: Simulation of automotive systems in the context of autosar. Technical report, Hasso-Plattner-Institut für Softwaresystemtechnik GmbH (2009)
11. : OSEK/VDX Communication. 3.0.3 edn. (July 2004)
12. AUTOSAR GbR: Requirements on Communication. 2.1.3 edn. (August 2008)
13. AUTOSAR GbR: Specification of Communication. 3.0.3 edn. (August 2008)
14. AUTOSAR GbR: Specification of PDU Router. 2.2.3 edn. (August 2008)
15. AUTOSAR GbR: Requirements on Gateway. 2.0.5 edn. (August 2008)
16. AUTOSAR GbR: Specification of CAN. 2.2.3 edn. (August 2008)
17. AUTOSAR GbR: Specification of I-PDU Multiplexer. 1.2.3 edn. (August 2008)
18. AUTOSAR GbR: Requirements on I-PDU Multiplexer. 1.0.5 edn. (August 2008)
19. AUTOSAR GbR: Specification of LIN Interface. 2.0.2 edn. (August 2008)
20. AUTOSAR GbR: Specification of FlexRay Transport Layer. 2.2.2 edn. (August 2008)
21. AUTOSAR GbR: Specification of CAN Transport Layer. 2.2.2 edn. (August 2008)

22. AUTOSAR GbR: Specification of Communication Manager. 2.0.2 edn. (August 2008)
23. AUTOSAR GbR: Specification of CAN State Manager. 1.0.2 edn. (August 2008)
24. AUTOSAR GbR: Specification of Generic Network. 1.0.2 edn. (August 2008)
25. AUTOSAR GbR: Specification of Diagnostic Communication Manager. 3.1.1 edn. (August 2008)
26. AUTOSAR GbR: Requirements on FlexRay. 2.0.5 edn. (August 2008)
27. AUTOSAR GbR: Specification of FlexRay Interface. 3.0.3 edn. (August 2008)
28. AUTOSAR GbR: Requirements on LIN. 1.1.4 edn. (August 2008)
29. AUTOSAR GbR: Requirements on CAN. 2.2.2 edn. (August 2008)
30. Huang, G., Bose, O., Patel, S.: Can controller area network. Technical report, Technische Universitt Berlin (June 2003)
31. AUTOSAR GbR: Specification of FlexRay Driver. 2.2.2 edn. (August 2008)
32. AUTOSAR GbR: Specification of LIN Driver. 1.2.2 edn. (August 2008)
33. AUTOSAR GbR: Specification of CAN. 3.0.3 edn. (August 2008)