Vehicle Diagnostic with AUTOSAR

H A N W U

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2008
Vehicle Diagnostic with AUTOSAR

H A N W U

Master’s Thesis in Computer Science (30 ECTS credits) at the School of Engineering Physics Royal Institute of Technology year 2008 Supervisor at CSC was Inge Frick Examiner was Stefan Arnborg

TRITA-CSC-E 2008:096 ISRN-KTH/CSC/E--08/096--SE ISSN-1653-5715

Royal Institute of Technology School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se
Vehicle Diagnostic with AUTOSAR

Abstract

AUTOSAR is the new standard software architecture design for automobiles, created in cooperation of the largest automobile manufactures. In this work a brief comparisisation between the CAN Transport Protocol specification in the new architecture and the existing standard ISO 15765 is made. And the CAN Transport Protocol module is implemented. An evaluation of the AUTOSAR in whole is made for future planning at SRE. It is learned that AUTOSAR is a competent and advanced architecture. It covers a wide area of automobile electronics and the specifications are well documented and easy to understand.

Bildiagnostik med AUTOSAR

Sammanfattning

# Table of Contents

Introduction ..................................................................................................................... 1  
Company profile ............................................................................................................. 2  
Problem Description ...................................................................................................... 3  
Theory .............................................................................................................................. 5  
  Incoming AUTOSAR ....................................................................................................... 5  
  Layers and BSW ............................................................................................................. 6  
  The world talks AUTOSAR .......................................................................................... 8  
  AUTOSAR Progress ...................................................................................................... 9  
The implementation ........................................................................................................ 10  
  Setting of company directive ....................................................................................... 10  
  Reading of AUTOSAR Specifications ......................................................................... 10  
  MISRA-C and requirements of programming ............................................................... 10  
  CAN – Controller Area Network ................................................................................. 11  
  CAN TP .......................................................................................................................... 11  
  USDT (ISO 15765-2) .................................................................................................... 12  
  CANTP SDU-ID .......................................................................................................... 12  
  CANTP Channels .......................................................................................................... 12  
  CANTP API and Expectations ...................................................................................... 13  
  Diagnostic Message Transmission ............................................................................... 14  
    Overview .................................................................................................................... 14  
    From PDU Router to CAN Interface ......................................................................... 14  
    From CAN Interface to PDU Router ......................................................................... 15  
  Drawbacks ..................................................................................................................... 15  
Evaluation of AUTOSAR ............................................................................................... 16  
Conclusion – Is tomorrow AUTOSAR? ......................................................................... 17  
References ....................................................................................................................... 19  
Appendix ......................................................................................................................... 20  
  State Diagram .............................................................................................................. 20  
  Tx FlowChart ............................................................................................................... 21
Rx Flow Chart ........................................................................................................................................... 22
Acronyms and abbreviations .................................................................................................................. 23
Introduction
The modern vehicle depends on hundreds of parts to function correctly. To keep those parts under the eye of a driver will make us feel safer. But how can we monitor these modules at the same time as we drive in traffic?

As we all know, all vehicles are equipped with an instrument panel which displays some information (turn on a red light, for example) as soon as something is wrong. That is vehicle diagnostic. With the help from built-in sensors, self-diagnosed ECUs\(^1\) and software, the vehicle can monitor its crucial functions and indicate problems to the driver through the instrument panel to avoid dangers on the road.

Stoneridge Electronics (SRE) has been a Tier one supplier of diagnostic platforms for a large number of vehicle manufacturers. During the 30 years of service they have created many different versions of diagnostic software. In the beginning when everything was fresh it was a challenging and fun task to create those programs. But as time passes by, it feels that it is crucial to gather them into one standard, both for better time and cost efficiency and for easier adaption into future projects. Until today it is still so unorganized in the automobile world that every time a new version of trucks comes out the existing program has to be heavily modified or sometimes new software has to be written.

The bright news comes from AUTOSAR\(^2\), an incoming software module standard developed by a group of OEM\(^3\) manufacturers and Tier one automotive suppliers with a goal to achieve modularity, scalability, transferability and re-usability of functions.

The aim of this thesis was to study today’s existing diagnostic platforms of SRE and research the possibility of an adaption to AUTOSAR, and to find out if a full scale development of AUTOSAR in their new products is profitable. The project was assigned to me and Jan Dakermanji from Mechatronics KTH and completed with a close cooperation between us. With a common background reading and structure setting, we have implemented different part of the software module. I focused on the PDU Router to CAN Interface transmission and Jan the CAN Interface to PDU Router transmission.

---

\(^1\) ECU : Electronic Control Unit.
\(^2\) AUTOSAR: Automotive Open System Architecture.
\(^3\) OEM: Original Equipment Manufacturer
Company profile
Stoneridge was founded 1965 in Warren Ohio, by D.M. Draime. [1]

Stoneridge Electronics AB was formerly Berifors AB [2], which was founded as a buyout of the automotive electronics division from Ericsson Radio Systems. From 1988 SRE became a supplier of electrical and electronic systems for the automotive, truck and bus market. The concentration on heavy vehicles allows them to become a strong competitor in the area.

These are some main products of Stoneridge Electronics:

**Instrument Cluster:**
These advanced instrument clusters combined with display panel for trucks, buses and tractors can view the states and request for simple diagnostic softwares from critical ECUs of the vehicle.

**Tachograph:**
This is an advanced device that combines the functions of clock and speedometer. Its function is to record, evaluate and control driving time and speed. This means that if the Tachograph is malfunctioning the vehicle can even refuse to start.

**ECU:**
In a modern vehicle these are the brains. They control all the complex functions in the car including the fuel injection, Antilock Braking System (ABS) and Electronic Stability Program (ESP).

**Switch:**
This is a control box for electrical units. It should not only send a control signal to systems such as electrical windows, mirrors, and door-locks, but also supply power to systems that require it.
**Problem Description**

As it is today, the automotive world is full of widely used standards, every manufacturer has their own idea of which standard their hardware should use. SRE puts in a lot of work adapting their already working software into each new project. That is not easy, because a new ECU always requires some patches in the software. What SRE does now is modify the codes and try to put them together running again. Since the code modules are heavily dependent on each other, it is a time consuming process to make modifications. This is creating problems for new workers who are assigned to a project, which slows the expansion of business and in the long turn threatens the survival of the company.

To solve this problem SRE wants to generalize their programs so that they can be easily ported to other new platforms. Since AUTOSAR is coming out with the same goal and with a strong team behind it, it seems profitable to use it in the future. The only problem is that AUTOSAR is so new that an evaluation is needed before any decision is made at SRE. A study of the AUTOSAR standard is therefore needed to determine if it can fit the needs of current SRE customers.

The picture below shows the software architecture design of one SRE diagnostics project today.

![Diagram of SRE software architecture](image)

*Figure 1: The SRE software architecture today*

---

1 From Stoneridge Electronics
Since the invention of the microcontroller the electronic revolution has affected all domains in our world. In the automotive industry, up to 90% of all innovations are attributed to Electrical Engineering. The electronic scheme and programs are becoming more and more complicated, as Figure 2 shows, increasing with exponential speed.

This will cause a quality impact once the required engineers surpass the existing manpower. Recently 50% of Volvo PV’s new car platform V70 and XC70 have been recalled to factory for modifications because of a program failure. This is just an example of the problem expected.

Because there is no general standard for the architecture, all the automotive manufactures are facing larger and larger cost for their software development; even then they are risking more bugs in their new produced vehicles. That is the reason for the manufacturers to gather together and try to make a new standard architecture, to shorten development time and reduce the cost in future.
**Theory**

**Incoming AUTOSAR**

The main goal for AUTOSAR is to enable plug-n-play for future cars. It is an architectural standard, defining clearly the interface between modules and between them and the RTE\(^1\). But it is not intended to define how the modules actually work. Most functions in a car already have their stable working versions of software based on existing standards. AUTOSAR is trying to build upon those standards and not to change them with a fresh architecture design (Figure 3).

![AUTOSAR architecture overview](image)

*Figure 3: AUTOSAR architecture overview*

Modularity, scalability, transferability and re-usability are the keywords of AUTOSAR\(^5\). These are made possible by defining a full function standardized interface for each module.

By modularizing the software elements, a manufacturer can easily choose the function modules according to the requirement from the customer: just put the modules with the desired functions into the software and it is ready for use, no need to worry how it will affect other parts of the system.

---

\(^1\) Run Time Environment
With scalability the combination of existing software modules in different platforms become possible because AUTOSAR only defines the interface, the complexity of the functions is flexible based on the needs of each vehicle type.

Transferability means that you can put the module where it is needed and where it will use the least system resources. This will ensure the optimization of resource usage throughout the vehicle.

With the architecture standard provided by AUTOSAR, re-usability of functions will be a fact; more time can then be put on the improvement of functions instead of developing new.

By creating a common interface between the modules and providing a RTE for the hardware-independent software layer, AUTOSAR is trying to unite the world of automotive software. Once the standard breaks through, all suppliers will talk the same language. So one manufacturer can choose different suppliers for their ECU platforms and they will still cooperate as if they are made by one, thus minimizing the cost for software.

Layers and BSW
The architecture of AUTOSAR divides the software modules into layers based on their functionality. Each layer contains several blocks that handle different tasks in the vehicle electronics. Figure 4 shows how the layers are connected up- and downwards.

![AUTOSAR Layers](image-url)
Applications are only connected to AUTOSAR through the RTE. The microcontroller abstraction layer (MAL) has the hardware functions like memory stacks, the I/O and the drivers, and ECU abstraction layer is built upon the MAL thus made independent of the hardware. Some of the functions have direct access to the hardware for RTE through System services and Complex drivers. This you can see more detailed in Figure 5 describing basic software modules.

**Figure 5 : Autosar Basic Software Modules**

Each service block contains several basic software modules (BSW). Currently AUTOSAR has defined 54 BSWs for vehicle electronics. Every BSW have a standardized interface but give free hands to the supplier to implement the internal function. This allows a scalable system.

AUTOSAR also provides specifications of tools for modeling and generating templates. Those can be found at their homepage http://www.autosar.org.
The world talks AUTOSAR
The discussions about the common objective were initially started by BMW, Bosch, Continental, DaimlerChrysler and Volkswagen in August 2002 and soon with Siemens VDO in their partnership. The core partners lately expanded with Ford Motor Company, Peugeot Citroën Automobiles S.A, Toyota Motor Corporation and at last in November 2004 with General Motors. These manufacturers were gathered to find a way to minimize the struggle for upgrading their software. It might be their future plan to unite their forces, like the uniting of the PC market.

![Member's status in July 2005](image)

By Oct 2006, there were 10 core partners, 52 premium members and 45 associate members in the AUTOSAR group. New members are still flowing in. The current members-status can be reviewed at AUTOSAR’s homepage. http://www.autosar.org

SRE has joined the membership at the end of 2007, this allow them free access to current specifications and other members software.
**AUTOSAR Progress**

AUTOSAR v2.1 specifications are used in this thesis work. A full RTE description will be available within version 3.0, which was planned to be released in Nov. 2007 but was delayed. However, since release 2.1 has already given us a clear description of how their modules cooperate with each other and several OEMs for example Vector already developed sets of AUTOSAR BSWs\(^1\), it is not a matter if a new version will come, but only when it will come. As the schedule for AUROSAR (Figure 7) describes, version 4.0 is planned to release at end of 2009. With that it will be possible to have complete test tools for evaluating AUTOSAR modules (conformance tests). BMW has planned to release a full AUTOSAR car in the year 2010.

![AUTOSAR top level schedule](image)

*Figure 7: AUTOSAR top level schedule*

No matter how the schedule is kept, the standardization of the automobile industry is coming true. And when it comes, AUTOSAR is probably the most mature and competitive standard that exists on the market.

---

\(^1\) BSW: Basic Software Module. A well interfaced module that fulfills some basic functionality.
The implementation

Setting of company directive
In the beginning the project was supposed to investigate the possibility of unifying the existing projects of SRE. If possible, an AUTOSAR compatible diagnostic platform should be created or at least started. Soon after the first study of the AUTOSAR specifications, we discovered that AUTOSAR was designed for unification so we might as well make it central to our project instead of defining our own methods. As most OEMs seem interested in AUTOSAR, SRE has no other choice than to adapt. If we go in our own direction it will generate two projects for the company which doubles the cost.

So the task was then to implement a module based on AUTOSAR specifications.

Since the vehicle diagnostic platform is complicated, we chose to limit our work to a simple part of it. Our goals was to understand the AUTOSAR basics and try to make a module that can be tested and proven to work, to increase SRE’s confidence in their ability to create a complete set of AUTOSAR BSW by their own.

The work was then decided to be an implementation of CANTP to evaluate and estimate the development process.

Reading of AUTOSAR Specifications
The specifications from AUTOSAR cover most areas of automobile electronics and are still expanding. The documents are well organized and thought through. Since we were going to work on the CANTP part of diagnostic, we focused our reading on the specifications of communication stacks.

MISRA-C and requirements of programming
MISRA is an abbreviation of Motor Industry Software Reliability Association. Our program is written in C and therefore we must follow the standard MISRA-C as required by SRE. Some of the standard functions for ANSI-C are not supported in MISRA-C, for example the memory allocation function malloc(), but otherwise we did not meet any difficulties in our usage of MISRA-C.

We needed also to write our code according to SRE guidelines for software development. This is mostly a requirement on the appearance of the program, i.e. the choice of function and variable names, the format of comments and a few restrictions about program structure. For example the guideline has advised against the usage of keyword return at the middle of a function, this was hard to follow sometimes but we did what we could to avoid its appearance. For functions and variables, AUTOSAR has
already defined most of the names in its specification and we have to meet AUTOSAR requirements first.

Finally, SRE also requires us to LINT our program. LINT is a nice tool to find out errors that are not generated in compile time.

**CAN – Controller Area Network**

CAN is a broadcast, differential serial bus standard for connecting electronic control units\(^1\). It was originally developed by Intel and Bosch, 1988. In this network every node (ECU) can receive and send messages at different times. Each message frame contains a header (29 bits or 11 bits depending on addressing format) and a message body (8 bytes). A message that is longer will have to be split into frames and in this case timing between each frame is important for the transmission to be successful. This can cause problems if more than one ECU need to send very often, but the problem is solved by creating several hardware channels to separate those talkative ECUs so that they can chat simultaneously.

**CAN TP**

Transport Protocol for CAN. Figure 8 shows where it is in a diagnostic platform. This is a module built on the standard ISO-15765. The function of this module is to segment a long diagnostic message from the PDU router to several CAN frames and carry them over to the CAN interface. Vice versa, it merges the CAN frames from the CAN interface into a complete diagnostic request and then transmit it to the PDU router. This is only for the CAN hardware, for other communication hardware AUTOSAR has defined corresponding transport protocols, for example FLEXRAYTP.

---

\(^1\) [http://en.wikipedia.org/wiki/Controller_Area_Network](http://en.wikipedia.org/wiki/Controller_Area_Network)
**USDT (ISO 15765-2)**

There is an existing module from Stoneridge based on ISO standard 15765-2 USDT (see figure 1). Since the specification of CANTP [6] from AUTOSAR is based on ISO-15765 [7], a comparison between the SRE implementation of USDT and CANTP is made in hope to give an easier understanding for developers from the company.

The USDT module from SRE is used in many of their diagnostic projects. The goal for us was to investigate the possibility of replacing it with the AUTOSAR CANTP module.

**CANTP SDU-ID**

A diagnostic request is often only a few bytes but the reply can sometimes reach kilobytes.

A CAN device transmits and receives diagnostic messages in frames of 8 bytes length. It is CANTPs task to keep track of the messages, divide them into frames, and merge them back again.

The AUTOSAR specification describes the ID translation for message tracking. Each SDU (N-SDU and L-SDU in Figure 9) is assigned a unique ID based on the source address, target address and message type and in extended addressing mode, the NTA value. This ID will be one of the parameters when calling API functions.

The AUTOSAR specification has not defined exactly how the ID will be computed when a message traverses different layers. The definitions of RxNsdu and TxNsdu\(^1\) in the configuration file maybe are the keys of how this actually operates.

**CANTP Channels**

Internal channels for transmission are defined in the AUTOSAR specification. Each SDUID is assigned to one channel, but a channel can be assigned by more than one SDUID. A share of a resource is therefore allowed in CANTP, the assignments are done in the configuration file. Each channel has a state indicator to avoid data collision and an internal buffer of 8 bytes length for temporary storage of message data. A Channel is locked when one SDUID is using it and unlocks when the whole transmission is complete or aborted due to error.

---

\(^1\) RxNsdu and TxNsdu are configuration constants for TX/RX sdu.
**CANTP API and Expectations**

The API functions that are defined in specifications are `CanTp_Init()`, `CanTp_GetVersionInfo()`, `CanTp_Shutdown()`, `CanTp_Transmit()`, `CanTp_MainFunction()`, declared in `CanTp.h`, call-back functions `CanTp_RxIndication()`, `CanTp_TxConfirmation()` declared in `CanTp_Cbk.h`.

The call-back functions are functions that are invoked by the upper- or lower-layer when they receive a transmission request from CanTp.

The interfaces that CANTP expects from other modules for fulfilling its core functions are `PduR_CanTpProvideTxBuffer()`, `PduR_CanTpProvideRxBuffer()`, `PduR_CanTpTxConfirmation()`, `PduR_CanTpRxIndication()`, `Dem_ReportErrorEvent()`. The file structure for CANTP is shown in Figure 10.

Simplified functions for these expected interfaces were implemented in an external file in the project. Thereby a test program were created that will run callback functions and send test messages through the CanTp module for debugging and evaluation.

**Figure 10: File Structures of CANTP according to CANTP Specification**
Diagnostic Message Transmission

Overview
Based on ISO 15765-2, the purposes of AUTOSAR CANTP are:

- Segment a long diagnostic message to frames.
- Transmit frames.
- Merge received frames into a diagnostic message.
- Handle errors.

Figure 11 shows an overview for how a transmission is carried out.

From PDU Router to CAN Interface
A CAN diagnostic request generated in the application layer is sent through the DCM (Diagnose Communication Manager) and redirected by PDUR to the CANTP module. A call to CanTp_Transmit() is made and CANTP records the SDUID and request type into the state registers (set the state to CANTP_TX at vector index SDUID) and return with value OK when all prerequisites are granted. A complete explanation of states is shown with the State Diagram in the appendix.

The tasked function CANTP_MainFunction() should be called by the OS timer. Because we do not have the OS environment of AUTOSAR we have chosen to use our test program to call it and in hope that when AUTOSAR OS is complete it will support a simple method for background tasks. CanTp_MainFunction() loops through the vector of state registers and calls appropriate callback-functions. When the state CANTP_TX is detected in the state registers, a call to an internal function that handles the transmission is made. In the internal function a buffer request to PDU-Router is made (PduR_CanTpProvideTxBuffer() ) and a buffer that contains the message for transmission is provided by the PDU-Router as a pointer, along with the length of the message, by returning a pointer to the message the need of memory is kept low. If the buffer is not available due to lack of system resources, a BUSY will be returned by the PDUR, the buffer request will then be resent in the next loop of the state vector, three consecutive BUSY will cause the send to fail and the state register is cleared and a confirmation with result NOT_OK is sent to PDUR (PduR_CanTpTxConfirmation() ). Here
the functionalities of the PDU-Router are defined in the AUTOSAR specification for the PDU-Router, which will not be explained further here.

If the message to be transmitted can fit in a single frame, a single frame (SF) transmission will be executed. Otherwise, CANTP will perform a multi frame transmission with first frame (FF) and consecutive frames (CF). This procedure is a complicated process; it has buffer handles, timing controls, flow-control feedbacks (FC) and confirmations (CONF). The transmission is described in the Tx Flowchart in the appendix.

**From CAN Interface to PDU Router**

A message that was sent from another ECU arrives to CANIF and thus triggers a call to the CANTP function CanTp_RxIndication(). This function will check if all prerequisites are met, that is if SDUID and its corresponding channel are free. Or in case a transmission with the same ID is already in progress, if the timer is not overdue and the frame sequence number is right. Then if necessary a call to PduR_CanTpProvideRxBuffer() is made to allocate a buffer for storage of message data. Flow-controls and confirmations are also involved in the transmission and the process is better explained in diagram Rx FlowChart in the appendix.

**Drawbacks**

Being an AUTOSAR module, CANTP is a forward leap for the standard. Many improvements are made compared to the earlier standard USDT, for example the buffer allocation method and the channel/ECU configuration. Our implementation has several drawbacks though:

Because a real AUTOSAR environment is not available, a reliable test is not possible to carry out. Therefore a test program based on Windows OS is made. That is, the CPU type is not the same as it will be in an automobile ECU, the RTE for our tests is Windows and not AUTOSAR OS. So if put in real hardware, the timing can cause the program to fail. The call to CanTp_MainFunction() which should be triggered by a timer needs to be adjusted. The problem with the test environment should be solved when AUTOSAR reaches version 4.0, in which the specifications of conformance tests are expected to be complete.

The USDT from SRE can handle full-duplex transmission. In the AUTOSAR specification, the CANTP only supports half-duplex; this means that one channel can only handle one direction of transmission. But this is the internal functionality of the CANTP. There is not any limitation for a full-duplex transmission of other modules through CANTP, giving them different SDUIDs.
At the end of the project there was a wish from SRE for a hardware implementation of CANTP. Because this will need an understanding and modification of the CAN driver, which we barely know yet, we did not have enough time for this.

**Evaluation of AUTOSAR**

The development of CANTP was not a hard task with all the support and specifications we got from Stoneridge and autosar.org, because it is really a small module in the diagnostic and the connection between CANTP and other modules are few, at the upper layer the PDU Router and at the lower layer the CAN Interface, plus a few calls to DEM for self-error reporting. As a new software developer in this area it should be quite simple to understand the requirements and methods of processing. The AUTOSAR documentation for CANTP is sufficient for completion of a program. The technical compatibility to previous USDT (ISO 15765) is high because CANTP is basically built on the same standard, but with much more detailed clarifications. In comparison to the existing USDT, details for transmission, buffer strategy, channel states, the interactions with other modules, and the error reports are all specifically defined in AUTOSAR – it is a full-scale standard instead of being only a transport protocol. So based on what I have met in the coding phase, I do not see any obvious obstacles for moving toward the new standardized software environment. The standard, if it continues to perfect itself, will become a major breakthrough for the automobile industry, anyone who wants to stay as a part of future development will have to move into this area sooner or later. The only question is how much will the cost be for each of them.
Conclusion – Is tomorrow AUTOSAR?
Based on AUTOSAR’s own vision and how the membership status is right now, we may reach a point that tells us what is good and bad.

**Good for car manufacturer:**
- The price for software will come down because of standardization. This is the major reason for most of the industrial standardizations.
- Service can be united, and the cost will be reduced further. This will also benefit end-users in a smaller scale.
- Common platforms can be interchanged. This reduces the time and effort of upgrades.

**Good for supplier:**
- Development time for new applications is shortened. Because most of the modules are standard lesser effort will be wasted on unclear interfaces and bug fixes.
- Less to worry about in architecture design. Because AUTOSAR already done it.

**Good for user:**
- Safer modules.
- Service and repair are cheaper. (Can choose 3rd party service depot)

**Bad for manufacturer:**
- Higher hardware requirement.
- Not yet clear standard. Does not know the outcome yet.

**Bad for suppliers:**
- They have no strategy for their future yet.
- Competition will come from every other supplier who was not in the business before.

**Bad for user:**
- Unknown.

**Decision:**
AUTOSAR uses existing standards in most of the modules for their functionality, even so it still has a survival problem, do the suppliers want to change their already fine tuned software, and turn toward an unknown future? There are some discords and uncertainties of the standard within suppliers and OEMs in the development phase[^8].
But AUTOSAR will keep its leading position of standardization and better tools for developing AUTOSAR software are coming out.

By the time of writing this report, AUTOSAR has given out several new documents at revision 3.0; many of them are in the conformance test area. This means that the whole architecture is maturing and going to be completed soon.

Although AUTOSAR is not designed for trucks and buses; although trucks usually go their own way of development, the AUTOSAR architecture is still leading in time, and with a minor modification/add-on of its BSWs it can be used in trucks as well.

So my proposition to SRE is: be prepared for the future standard.
References

1. About Stoneridge
2. About Stoneridge Electronics
   http://www.stoneridge-electronics.com/index.html
4. Dagensnyheter
   http://www.dn.se/DNet/jsp/polopoly.jsp?d=1064&a=717749&rss=678
5. Homepage of AUTOSAR
   http://www.autosar.org
6. CANTP specification
7. ISO standard
   mber=33616
8. Autosar standard suffers discord - Christoph Hammerschmidt
   http://www.eetindia.co.in/ART_8800472235_1800007_NT_b09ddcdf.HTM
Tx FlowChart

1. **PDU -> CAN TP**
   - CanTo_Transmit(DLen)

2. **Check State**
   - IDState[CANF, TX] = ChannelState(Chr=TX, PROCESSING)
   - If not, return NOT_OK
   - Main Function: Tasked Process
     - Loop W/ ID
       - IDState = CANF, TX
         - If not, return NOT_OK
       - Pdu_FrmIDBuffer(ID) ≠ null
         - If OK, return OK
         - If BUSY, return NOT_OK
       - Send_Pkt(ID, CANF, TX)
         - If OK, return OK
         - If FAILED, return FAILED

3. **Send Data**
   - IDState = IDLE
     - PduR_TxConfirm(OK)
       - IDState[CANF, TX] = ChannelState(Chr=TX, IDLE)

4. **PduR_TxConfirm(OK)**
   - IDState[CANF, TX] = ChannelState(Chr=TX, IDLE)
   - IDState = IDLE

5. **Task Data ID = Task Catalog Length?**
   - If NO, return NOT_OK
   - IDState = IDLE
   - If YES, return OK

6. **CanF Confirmation**
   - IDState = IDLE
   - PduR Request
     - IDState = CANF, TX
       - Transmit Success
         - IDState = TXCONF

7. **ID State = IDLE**
   - PduR Request
     - IDState = CANF, TX
       - Transmit Success
         - IDState = TXCONF

8. **ID State = TXCONF**
   - PduR Request
     - IDState = CANF, TX
       - Transmit Success
         - IDState = TXCONF

9. **ID State = IDLE**
   - PduR Request
     - IDState = CANF, TX
       - Transmit Success
         - IDState = TXCONF

10. **ID State = TXCONF**
    - PduR Request
      - IDState = CANF, TX
        - Transmit Success
          - IDState = TXCONF

11. **ID State = IDLE**
    - PduR Request
      - IDState = CANF, TX
        - Transmit Success
          - IDState = TXCONF

12. **ID State = TXCONF**
    - PduR Request
      - IDState = CANF, TX
        - Transmit Success
          - IDState = TXCONF

13. **ID State = IDLE**
    - PduR Request
      - IDState = CANF, TX
        - Transmit Success
          - IDState = TXCONF

14. **ID State = TXCONF**
    - PduR Request
      - IDState = CANF, TX
        - Transmit Success
          - IDState = TXCONF
Rx Flow Chart

- CallToRxIndication(I, Data)
  - IPC == FC?
    - No
    - ChannelState[i] = TX || (ChannelState[i] = RX & ChannelState[j] = IDLE)
      - NO
      - ChannelState[i] = RX
        - SF == FF
          - NO
          - Phy_PeuRtiication(I, NOT_OK)
        
        - YES
        - SetStat(I, ChannelState[i] = RX & ChannelState[j] = Busy)
          - SF? FF
          - SF == FF
            - NO
            - SetTaskOutLength
              - RxAccess(I)
        
        - YES
        - IdStateSegmentInProgress == 1 & CFC Count = PCL/CF
          - YES
          - ClearState (ChannelState[i] = IDLE & ChannelState[j] = IDLE)
        
        - NO
          - If State SegmentInProgress == 1 & CFC Count = PCL/CF
            - SetStat(I, ChannelState[i] = RX & ChannelState[j] = Busy)
              - SF? FF
              - SF == FF
                - NO
                - SetTaskOutLength
                  - RxAccess(I)
            
            - YES
            - ClearState (ChannelState[i] = IDLE & ChannelState[j] = IDLE)
# Acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSAR</td>
<td>Automotive Open System Architecture.</td>
</tr>
<tr>
<td>BSW</td>
<td>Basic Software Module.</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network.</td>
</tr>
<tr>
<td>CanIf</td>
<td>CAN Interface.</td>
</tr>
<tr>
<td>CANTP</td>
<td>CAN Transport Protocol.</td>
</tr>
<tr>
<td>DCM</td>
<td>Diagnostic Communication Manager module.</td>
</tr>
<tr>
<td>DEM</td>
<td>Diagnostic Event Manager.</td>
</tr>
<tr>
<td>ECU</td>
<td>Electronic Control Unit.</td>
</tr>
<tr>
<td>Full-duplex</td>
<td>Point-to-point communication between two nodes is possible in both directions at one time.</td>
</tr>
<tr>
<td>Half-duplex</td>
<td>Point-to-point communication between two nodes is only possible in one direction at a time.</td>
</tr>
<tr>
<td>CAN L-SDU</td>
<td>SDU of the CAN Interface module. It is similar to N-PDU but from the CAN Interface module point of view.</td>
</tr>
<tr>
<td>MAL</td>
<td>Microcontroller Abstraction Layer.</td>
</tr>
<tr>
<td>CAN N-SDU</td>
<td>SDU of the CAN Transport layer. In the AUTOSAR architecture, it is a set of data coming from the PDU Router.</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer.</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit.</td>
</tr>
<tr>
<td>PduR</td>
<td>PDU Router.</td>
</tr>
<tr>
<td>RTE</td>
<td>Run Time Environment.</td>
</tr>
<tr>
<td>SDU</td>
<td>Service Data Unit.</td>
</tr>
<tr>
<td>SRE</td>
<td>Stoneridge Electronics.</td>
</tr>
<tr>
<td>USDT</td>
<td>Unacknowledged Segmented Data Transfer.</td>
</tr>
</tbody>
</table>