CHALMERS

Simulation based verification for LIN /CAN application layer

Master of Science thesis

Afshin Etemad

Department of Signals & Systems
Division of Signal Processing, Biomedical Engineering and MPCOM.
Chalmers University of Technology
SE-412 96, Gothenburg, Sweden 2016

Master's Thesis 2016:4

Simulation based verification for LIN /CAN application layer

Department of Signals and Systems
Chalmers University of Technology
SE-412 69 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Simulation based verification for LIN /CAN application layer
Afshin Etemad
Department of Signals and Systems
Chalmers University of Technology

Abstract

The V-model can be described as the conceptual and graphical illustration of a software development lifecycle. The left and right side of the V represent the steps in software lifecycle development, from the requirement decomposition phase until the function verification phase. This V-shaped model is a widely accepted strategy in developing electrical modules in automotive industry. A single loop in V-model starts with development and completes with verification. After several loops, the software will be mature enough for final verifications.

This thesis addresses the issues that come with V-model on developing a system with one master node and multiple slaves. Master-Slave is a standard of communication in which one part has unidirectional control over the other part. This thesis suggests a verification routine with MATLAB-CANoe interface on CAN/LIN modules. This study presents how this method can minimize the functional defects and signalling problems. On this study, a Simulink model for the CAN node is done. The Simulink model is verified on the test environment with the physical slave node. The result of the study is presented as an approach to improve the quality process at Volvo Cars.

Keywords

Power Seat Module, Seat Comfort Module, LIN slave, CAN master, simulation based verification, CANoe panel, V shape development strategy,

Acknowledgements

I would like to thank Volvo's research and development department in Gothenburg and all the colleagues in the office. Particular Jan-Erik Henriksson and Parvaneh Parhizkari who provide background and detail for Volvo's components and helped to learn this much. I would also like to thank Britt Gruvesäter for making this thesis possible and for letting me continue learning in Volvo. I am grateful for the assistance given by Srikar Muppirisetty and also for support and guidance of my examiner Henk Wymeersch at Chalmers University of Technology.

Afshin Etemad, Gothenburg

Table of Contents

Abbreviation

DUT device under test

PSM Power Seat Module

SCM Seat Comfort Module

HWIL Hardware in loop

API Application Programming Interface

AUTOSAR Automotive Open System Architecture

BSW Basic Software

CAN Controller Area Network

LIN Local Interconnect Network

ECU Electronic Control Unit

ISR Interrupt Service Routine

OS Operating System

RTE Run-Time Environment

VCC Volvo Car Company

SDU Service Data Unit

SWC Software Component

LC logical component

MOST Media Oriented Systems Transport

TDMA Time Division Multiple Access

SOP Start of production

RC resistors and capacitors

IL Interaction layer

TP transport protocol

NM network management

CAPL CAN Access Programming Language

1. Introduction

Automotive industry has faced rapid convergence of telecommunication and infotainment and multimedia in which brings vast variety of functions and services to the user. This flow of information imposes inevitable complication to the electrical modules. A full functional car comes with more than a hundred electrical modules in which the majority of them communicate through LIN/CAN (Local Interconnect Network) (Controller Area Network) protocol while the major modules using flexray and Ethernet. V-shape plan is predominantly used to develop and verify the LIN/CAN modules. ON V-shape plan one wing represents the development and other wing represents the verification phase. The focus of this thesis is to optimize the verification time for LIN/CAN modules, this is done by simulation based verification for LIN /CAN application layer.

1.1. Aim and objectives

This thesis aims to propose a solution to shorten the integration and validation phase when one the nodes is not physically available. The proposed solution is especially helpful when dealing with LIN/CAN modules with different SW maturity levels. Maturity level is a critical issue since if two components in a system has different software maturity levels then components cannot communicate with each other and therefore the system cannot be verified as a whole. The proposed solution is using integrated MATLAB Simulink shell model in CANoe simulation environment. The section Theory in this thesis is to describe this model. The suggested solution is started by developing the shell model for the DUT (device under test). In the next step, the C-code is generated from the shell model and imported in the CANoe environment to perform the integration test along with the physical slave node. The DUT in this study is PSM (Power Seat Module) and is used in the latest Volvo car, XC90. This module is a CAN master node and has a slave module called SCM (Seat Comfort Module).

2. Background

Volvo Group's Vision 202020 is about higher quality, leaner production and less cost with transformation. Vision 202020 in the new Volvo's guiding principle to which VCC had a new objective to reach a 20-month project time by the year 2020. A shorter development time provides higher product quality and a better process, also less re-planning and internal costs [1]. For most of the OEMs in the car industry, due to increasing complexity in electrical modules, in-house development of all components in vehicle is not an option.

In Volvo cars external suppliers develop the software and signaling solution for the whole or part of the systems. In fact, the majority of the systems are partly developed by one supplier, and partly in-house or by another supplier. Timing issues, syncing the maturity levels and verification of such systems are some of the challenges during the system lifecycle [1] .

3. Method

This chapter contains the methods used to gather the information and to understand the problem and also how the proposed solution is developed.

3.1. LIN & CAN protocol studies

In order to get the necessary background information about the nodes, an initial study about the LIN and CAN protocol is done. Initial information is gathered about the slave nodes LIN protocol [2] and also how the signaling has defined. Afterwards, a study is done about the details of transmission with master node. This information is gathered from the LIN Specification Package document [2], and also through the discussions with the VCC's LIN experts and from the LIN description files released by the VCC's signal database group .

3.2. Network topology and verification method analysis

In order to further development of the study, an assessment of the process flow and the verification methods for the LIN-CAN network is conducted. Result of this part of the study provides background to identify the limitations of the traditional verification methods. Finally and for further analysis, the optimization of verification methods is used for implementation.

3.3. Implementation

Focus of this part is on the setup using the LIN slave and the CAN master node. The Slave node is Seat Comfort Module and the master node is Power Seat Module. Power Seat Module is the standard module for all Volvo cars on the SPA platform. This is a comfort module to adjust the height, length, depth, and width of the seats. The low variant components are available on cheaper price but only provide basic functionalities. For the SPA platform on VCC, this module has been implemented by the supplier, through sending requirements and defining signaling and functionality by VCC. Development process started from the beginning of the SPA sourcing and did not finalized until the SOP (Start of production) which is significantly more than 20 months that is the VCC goal for leaner development time. One important point discovered in this phase of the study is that verification of the software issues and functionalities is a blocking factor that takes significant part of the development time.

In this phase of the study, the review over the PSM functionalities is conducted. Then a method to verify the CAN-LIN module is presented. This method uses the CANoe panels along with the Simulink shell models. CANoe is the major verification tool that is used in the VCC. This is a powerful tool that gives possibilities to tweak the signal values in real-time and observe the results on physical node. All or part of the network can be simulated by the CANoe. In this environment a simulated node can communicate with a physical node. Another node in this study is the Seat Comfort Module. Seat Comfort Module provides

massage on high variant and lumbar adjustment on low variants. This module is an Off-the-shelf node that has been ready for system verification a year before PSM's ready date.

The major purpose of this study is to evaluate system verification of SCM, an Off-the-shelf node, using simulated PSM. This approach can theoretically circumvents the timing gap that is caused by PSM development. In the last part, a number of test cases is verified using the proposed solution. Advantages and disadvantages of this solution is also presented in the last part.

4. Problem description

This chapter describes the motivation behind the study and how the proposed solution can increase the verification quality.

4.1. LIN/CAN system development

As it is described, a system solution based on the V model's lifecycle starts from the requirement decomposition. During this phase the complexity of the requirements can cause discrepancy between the developed functionality in different modules on the same system. These discrepancies can get escalated to Functional defects in the next phase of the development.
Message errors and synchronization problems are the common issues towards the development of the CAN/LIN application layer.
Since in the both protocoles, the signal datatypes shall have the same properties on the both sender and the receiver parts [3]. As these two modules are developed by two separate teams, signal property discrepancies occurs and can get detected not sooner than very late phase of system verification. A simple issue from this sort can be prevented during the development phase but will be extremely complicated to resolve during the mass production. Therefore, complex testing setups have now became an important part in the development of such communication systems. In order to meet the criteria of qualitative verification, different methodologies of systematic verifications is developed. Common goal of all testing methods is to verify the defects on earliest development stage. This thesis is also aimed to introduce a solution to accelerate the verification phase for the LIN/CAN system when the master node is not physically available [4]. In the conventional verification methods, the verification starts not before all nodes are physically available and can communicate with each other.

4.2. V-Model

The V-shape model for development and verification of CAN/LIN nodes is used for many years in the automotive industry. The main concern in this model is on the dramatic increase of cost and complexities when problems occur on the latest parts of the verification phase [5].

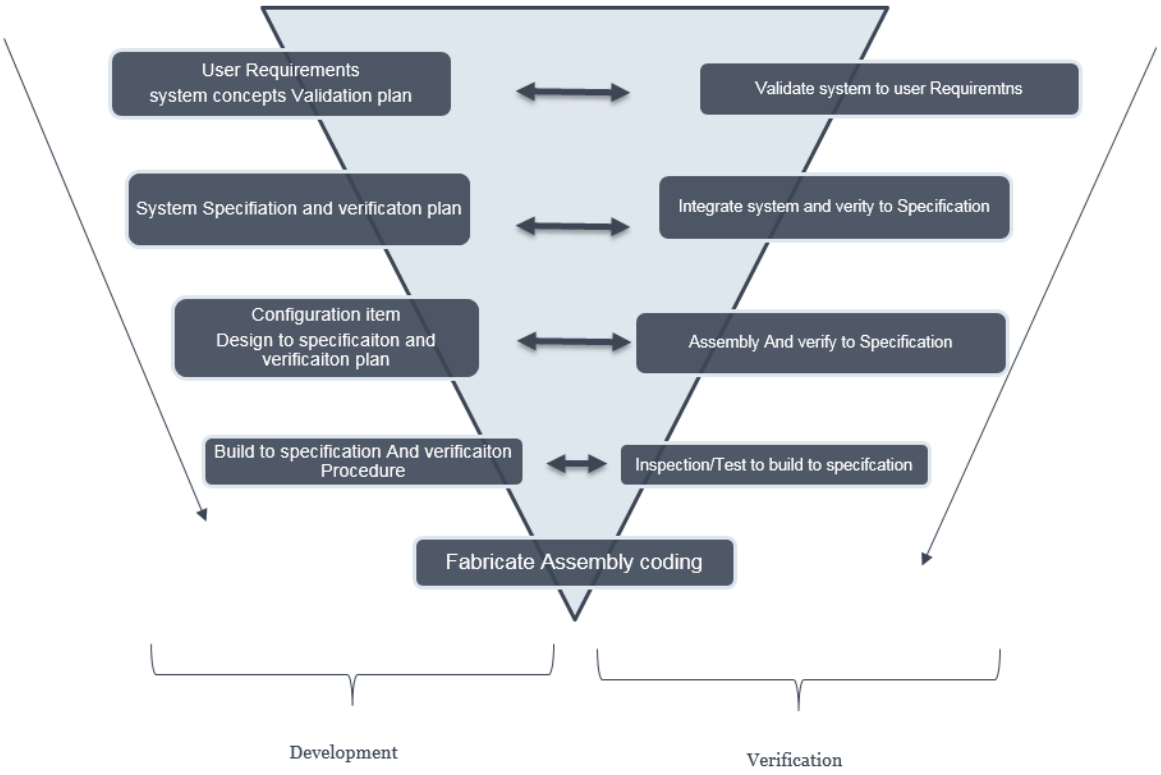

Figure 1. ***Original V-Model*** *[6]*

As can be seen in Figure 1 , the V/model starts from upper left in the requirement decomposition level (sent to supplier) and will be validated by the acceptance test in the right wing. These requirements creates the system requirement model and is verified on the integration system in the left wing. When basic functionality is implemented on architectural model, then the verification phase continues on subsystem model. The last phase is component design level that is verified by the HW-SW integration. If the verification lead to successful integration unit then design model can get started [5].

The motioned V-shape model describes the single component development apart from other affiliated and subordinated ECUs (Electronic Control Unit). In reality, SW development for ECUs on the automotive industry, a single component is always dependent upon the functionality of the master or slave node. Without comparative validation it would not be possible to have an up-and-running system. As Figure 2 shows, when two or more components (single master-multiple slaves) are on the model at the same time, this model repeats itself in ECU level and builds one system in the end.

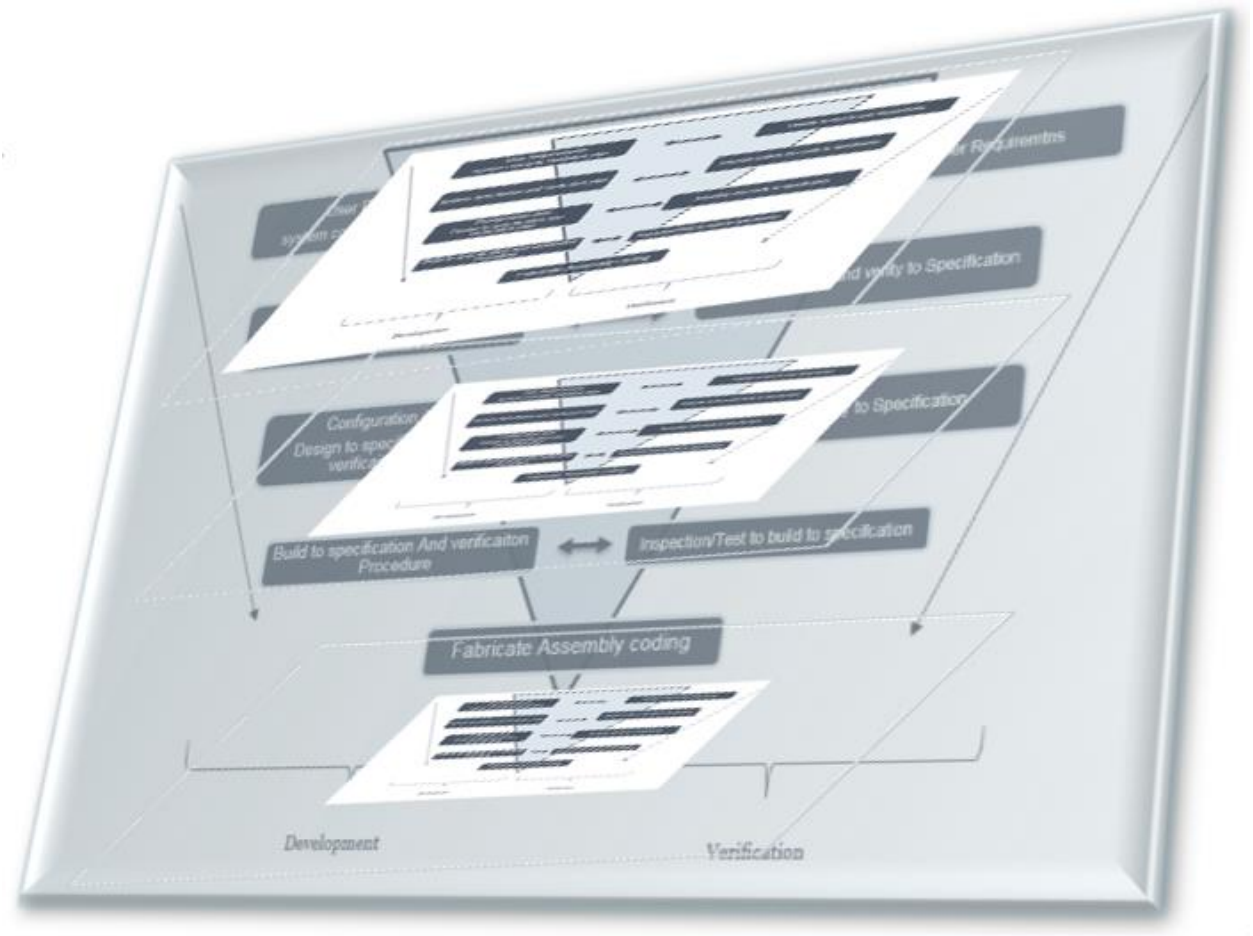

Figure 2 . Dual V-Model

Practically, applying dual or triple V-model for LIN/CAN system comes with delaying factor. System model verification shall be done on presence of all dependent components. Especially for the LIN/CAN systems the HW-SW integration cannot cover all layers of functionality when either master or slave is not involved.

4.3. Sync problems with V-Model for HWIL (Hardware in loop)

Realization in the first wing of the V model is done according to SW plans. SW plans are generically consist of different SW gates in which maturity level increases over time. As dealing with a standard LIN/CAN system consist of single master node and multiple slaves, full synchronization on SW maturity level between Masters and slaves is necessary. If the master node has 90% design intent and is ready to test on the unit design model while slave node is not even as nearly mature, then the dual/triple V-model fails to cover the integration test. SW maturity synchronization between two or more components in a LIN/CAN system is a usual problem when components are developed by different suppliers and has different complexity level [7] [8] .

It’s also usual that either of the nodes (master or slave) are bought from supplier in an early stage of the project as an Off-the-shelf solution. While the other node under development and following the SW plan. In that case a gap in the SW maturity synchronization grows future more. Integration and HIL (Hardware-in-the-loop) test can be delayed for months until both master and slave reach the same maturity level.

5. Theory

This chapter contains the theory behind the proposed solution. Starting by a comparison between LIN and CAN protocol, and also network topology used in the LIN and CAN nodes. In the next parts, Layer structure and Schematic layers of the CAN and LIN nodes and also detailed information regarding memory handling and actuators of the PSM along with a description for the MATLAB Simulink model and CANoe panel is presented

5.1. LIN vs CAN protocol

LIN is a serial communication protocol made for the connection of low to medium cost components with the non-complex sensor-actuator functionalities. Ideas over the LIN communication protocol is initiated by the LIN consortium, which is the collaboration between major car manufacturers. LIN consortium is the major entity to define the rule and regulation regarding the LIN protocol standards. This set of standards are getting frequently released on the LIN Specification Package (current version 2.1). LIN spec package covers the specification on the physical layer, transport layer, application layer and also the detailed information regarding tools and interfaces [2].

Figure 3.A LIN network [9]

As Figure 3 shows, LIN is a mono master network that supports up to 16 slaves using single wire bus. This protocol supports up to 20 Kbits/s transmission rate. LIN is self-synced by the master node and can detect errors using an 8 bits checksum and two parity bits. LIN can be used to build a serial communication, since master node syncs the clock therefore the slave does not need highly accurate clock and can use cheaper RC (resistors and capacitors) cell instead. Drivers that run with LIN are easy to use, non-complex and also available in the market [9]. These drivers require less harness usage and more reliable for the mid-range modules. The mentioned LIN network simplicities facilitate the development of extensions for functionalities on the components [9]. In the systems that require high-bitrate, robust transmission reliability and also has signalling with higher safety integrity level, the LIN protocol cannot afford the high fault rate [9]. Consequently, in such systems LIN protocols is not reliable enough and other protocols has to be employed. The following chart shows the speed comparison between LIN and other protocols.

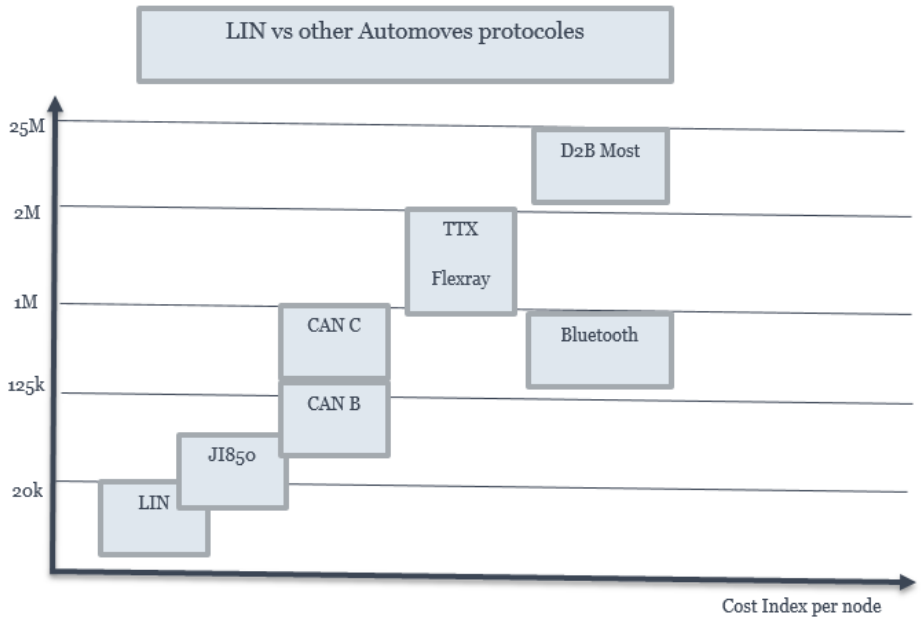

Figure 4. Speed comparison between LIN and other protocols. [9]

Despite the all advantages with the LIN protocol, the most well-known multiplex system used in the automotive industry is CAN protocol. The CAN bus has developed by BOSCH Company. This protocol is a multi-master, message broadcast transmission protocols. As shows in Figure 4, the maximum Bit-rate in this protocol is significantly higher than the LIN protocol and is up to 1 Mbps [10]. The most significant difference between the CAN and the LIN protocol is the transmission strategy. CAN unlike LIN does not use the master-supervised point to point transition. In CAN network all the nodes can broadcast their messages throughout the entire network and each node connected to the CAN network is able to receive any propagated signal [11] [12].

5.2. Network topology

Figure 5 depicts a LIN network made of several LIN slaves and one CAN master. The CAN master gateways the LIN signals to the other networks. The signals can get received by other CAN master and gateways back to their LIN slave. In this way, two LIN modules in two different networks can communicate with each other.

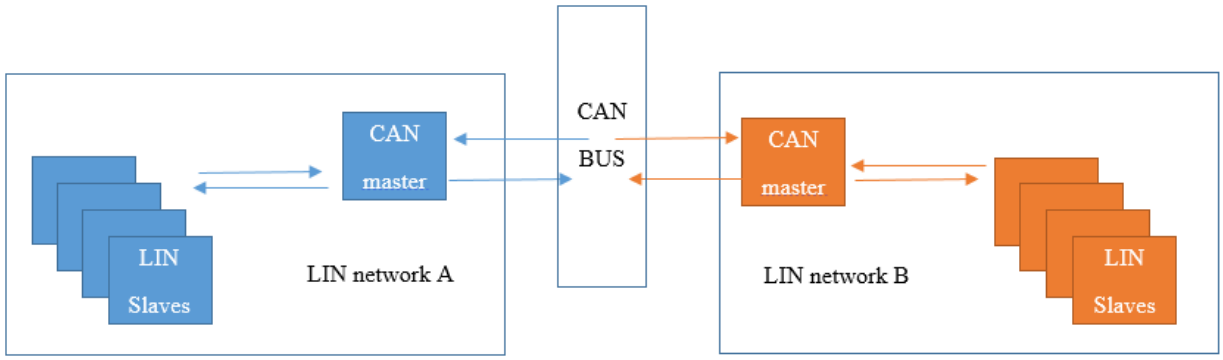

Figure 5. Signal broadcasting between two LIN networks

LIN protocol is not made to be used instead of CAN but rather a supplementary service to be used alongside the CAN. Nodes with strict fault-tolerance and high bandwidth transmission (>10 Mbit/s) will use other protocols such as flexray and MOST (Media Oriented Systems Transport). Flexray provides the time-event triggered functionality. This functionality is the main Flexray advantage towards the CAN. The CAN protocol is merely an event-triggered protocol and nodes can get access based on their priority. On flexray, the transmission time and frequency is defined in detail, using TDMA (Time Division Multiple Access) solution. In this solution, a message receives a specific time to communicate and the messages have timing hierarchy to get the access [7] [13] .

5.2.1 AUTOSAR Layer structure

AUTOSAR stands for AUTomotive Open System Architecture. AUTOSAR is the standard software architecture made by automotive suppliers/developer and OEMs for the automotive industry. AUTOSAR provides the common ground for different functions and platform and makes possible the integration from different OEM and supplier [14]**.** AUTOSAR architecture offers the modularity and gives OEMs the choice to specify the software based on the requirements. This standard also prevents the software redundancy when the same software is used on different platforms. In order to achieve flexibility that is mentioned, AUTOSAR architecture requires different layers of the program. Each part interacts with the inner layer or the outer hardware.

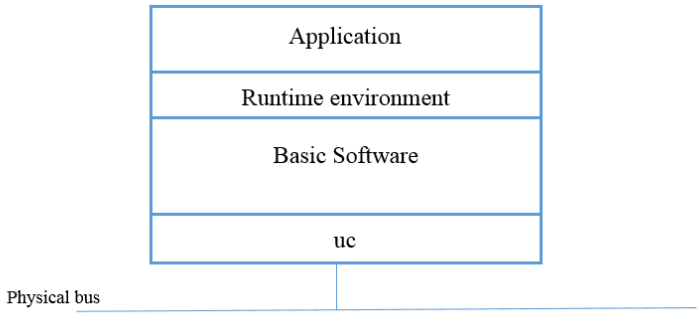

Figure 6. AUTOSAR software stack layers

As Figure 6 shows, AUTOSAR software stack has three different layers: Application Layer, RTE (Run-Time Environment), and Basic Software.

Application is the front layer of software which handles the functionality expected from the module. Inter module communication and also Intra module communication is managed by RTE. According to this setup, RTE get adjusted to fit the different types of microcontroller and ECU, while application layer shall be intact to be used in different modules. Basic software is made of different layers, such as:

- Abstraction layer, makes application layer even more modular.
- Service layer, contributes to memory handling, diagnostic services.
- Microcontroller abstraction layer
- Complex derivers

5.2.2 AUTOSAR software stack for the simulation ECU

The layers of AUTOSAR software stack can get further divided to the functional model. Based on the AUTOSAR guideline, the application layer is made of SWCs (software component). Each SWC represents a functionality provided by the ECU.

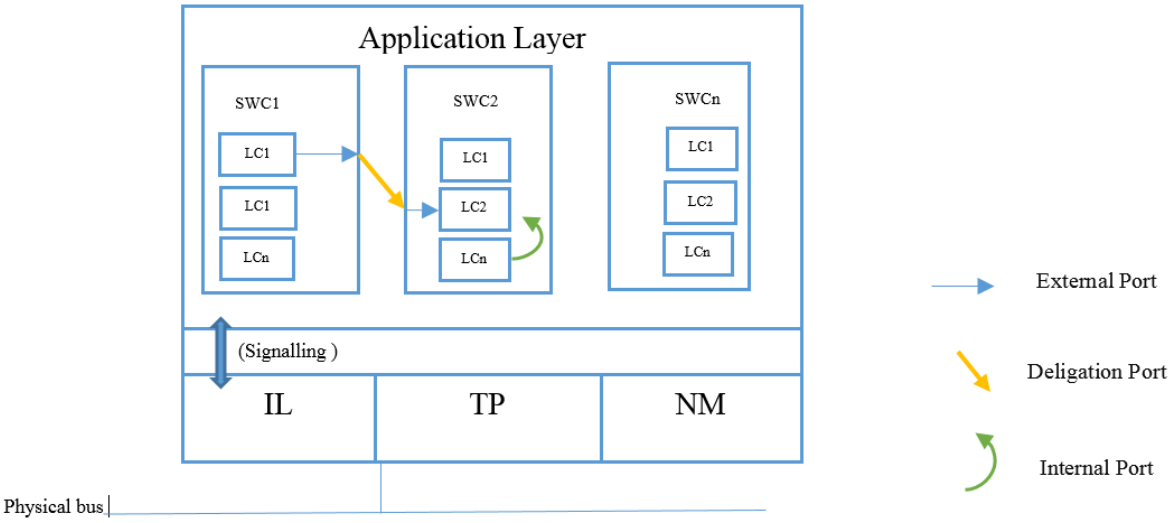

Figure 7. LC Ports on application layer of the ECU

A. AL (Application layer)

Application layer is the layer representing the external functionality of the component. These functionalities are defined by the system designer and the behavior in this case is described by the Simulink models and the state-flow. Each LC (logical component) represents a logical component and carries the signaling and the requirements from system point of view. LCs can get allocated in multiple ECUs and vice versa

(multiple ECUs realize functionality that is required from a single LC). As seen in Figure 7, each LC has a number of signals (external ports). In order to provide the functionality these ports shall get connected to another LC. If these LCs represent the different functionalities then these shall get delegated in the different SWCs. Therefore these ports get propagated out of the SWCs and get connected through so-called delegation ports.

B. IL (Interaction layer), TP (transport protocol) NM (network management)

Different software layers can be identified in a LIN node. These layers can generally be specified as the Interaction layer (responsible for the data transmission functionality), the transport protocol (handles diagnostic) and the network management. The interaction between these two layers get defined via signaling. This means that the network management is not a concern in the application layer or on the Simulink models.

Conventional System verification

Product lifecycle management (PLM) is the administration process for lifecycle of a module from the requirement phase until the manufacturing. As Figure 8 shows, this concept is the basis of the system architecture used in PLM development program. This concept gives flexibility to function or system designers to allocate the logical components on the appropriate ECU and also to verify implementation on the system level [15]. Requirement documents can be generated in each level and verification starts from the LCs and completes in the function level.

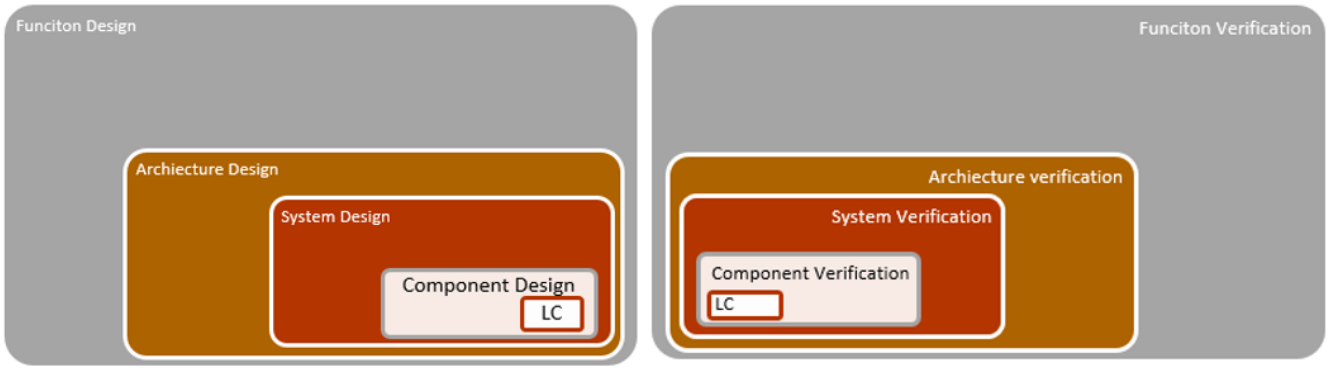

Figure 8. System architecture blocks from logical component to function

5.3. DUT description -Power Seat Module

PSMD is the module tailored to deliver the solution for the seat positioning on the driver and passenger seats for Volvo cars on SPA platform. This module handles the maneuvering of headrest, backrest and cushion extension. Handling of mentioned functionalities requires extensive signaling interface and also interaction with three motors and actuators. For this reason, Power Seat Module is designed to communicate using CAN protocoles. Seat functionalities such as lumbar and massage can either get handled by PSM or with pneumatic or electronic massage -comfort node which is derived by the LIN slave of PSM. In order to keep the modularity of system solution, these two functionalities are preferred to be handled in two different nodes. Figure 9 shows the drivers and interfaces of PSM. High side driver is the connection between the load and the power supply. Load is connected between the driver output and the ground. HSD (High Side Driver) gets activated when the output voltage is equal to the power supply. Apposite to the low side driver that gets activated as the output voltage is equal to the ground. MD (Motor Driver) is the driver that is responsible to activate the motors. This driver specifies how each of the motors get derived by the hall sensors. Pause setting to start the motors are also defined in the MD. The signaling shows where the system solution is defined on the application layer. The Comm layer redirects the signaling through controller and motor drivers. Transceivers sync the timing of bits and regulate the output voltage of the LIN and the CAN network to the level that is defined on the module [16].

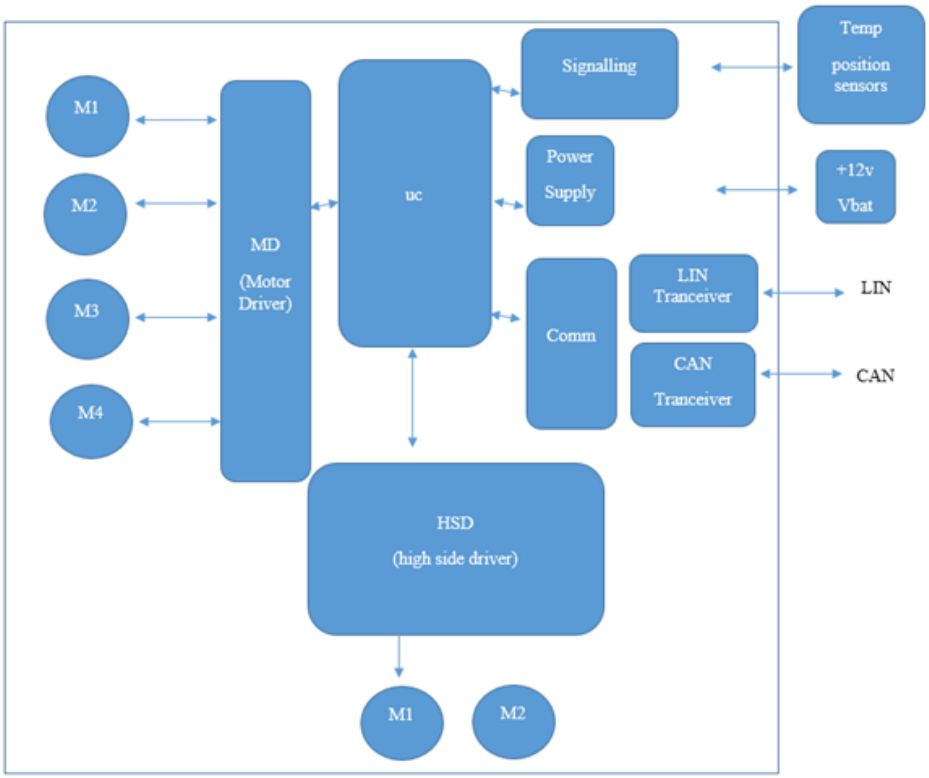

Figure 9. PSM motors and drivers interface

5.3.1 Switches and Commands

Seat commands can either get transmitted from the seat switches on the door module or remotely from other nodes like the HMI or the CEM. In the first case a block called Switch Interpreter filters the received signal commands. If two switch commands get received by the module then the whole movement shall get stopped immediately.

5.3.2 Motor actuators

If component is low or high variant, this module provides sets of motors. Each motor shall handle the movement in following angles:

Height up and downs

Front up and down

Slide forward and backward

Back inclination forward and backward

Lumbar Extend forward and backward

Lumbar height up and down

Seat cushion extension and shorten

Figure 10 shows the commands for the movement process where the sensors constantly return the actual position to the application level and the switches or the remote commands is processed by the application software and forwarded to the actuators. New position is written over the actual position on a loop-like process and movement continues to the block-position.

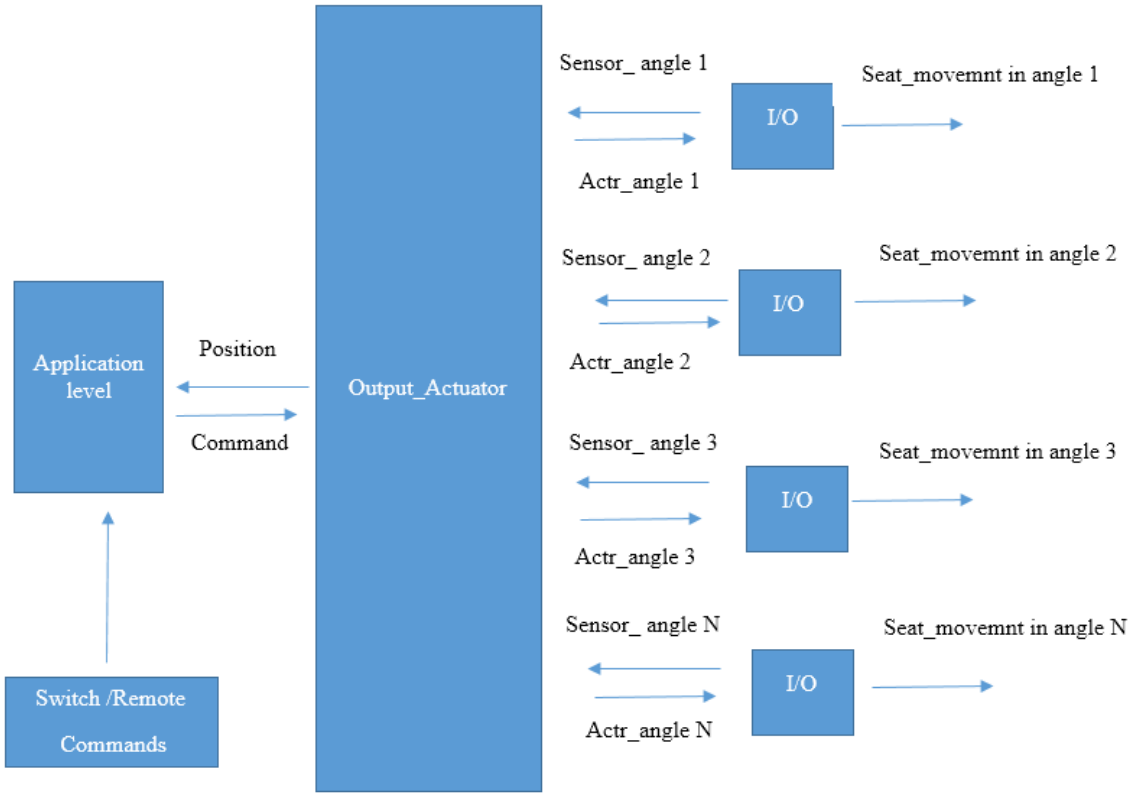

Figure 10. Command for movement process

As mentioned before, actuators can get activated through the resistor coded switches. The specified voltage range for hysteresis is done in parameterization or with remote CAN commands by the ECUs connected to the network.

5.4. Functional Mode of DUT

Figure 11 shows the logic and defines the transitions between four states using in Seat Module. These states characterize the different functional modes provided by the module.

1. **Normal State**: Which the only single-handed command communicated to the ECU, this mode is considered by the application level as an idle state.

2. **Stopping state**: Application software has the logic to prioritize the movement and to deactivate the actuator if simultaneous commands get received by the ECU.

3. **Manual state**: Application level can also move to this state as the remote commands are receiving from the other ECUs.

4. **Setting state**: Handles the memory functionality and the adjustment properties that is saved by the user.

Transition between the different modes is done consistently if the trigger condition 1-6 is true, otherwise the ECU will stays on the current mode.

Condition 1: IF Actuator_activatied =0

Condition 2: IF Button_Pressed ≠ 0

Condition 3: IF S1 =default: Actuator_activatied =off

Condition 4: IF S-requested=S4

Condition 5: IF number of incoming switch commands = X, X>1

IF Button_Pressed ≠ 0

IF Actuator_Antinpinch_flag ≠ 0

Condition 6: IF Actuator_activatied = 0

Condition 7: IF number of incoming switch commands = X, X>1

IF Actuator_Antinpinch_flag ≠ 0

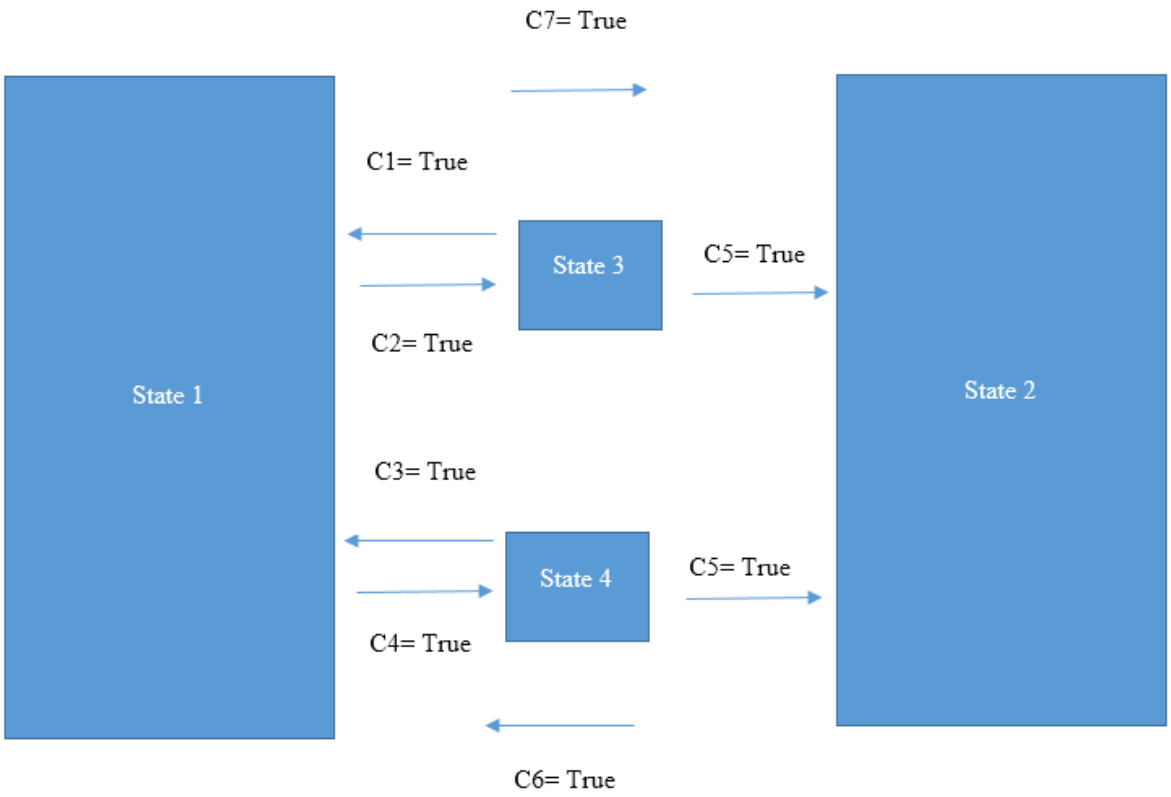

Figure 11. Transition between different states of PSM

5.4.1 Non-Volatile Memory handling

As Figure 12 shows, recalling and storing seat position can be performed by the user through HMI (Human Machine Interface) or seat switches. In order to store/restore the position, the current position shall be written over the selected profiled and then get stored on the corresponding part of the NV_RAM. Restoring operation is done using the same logic but in the opposite direction, from NV_RAM to the actuators while the ECU is set to setting mode requested by the user.

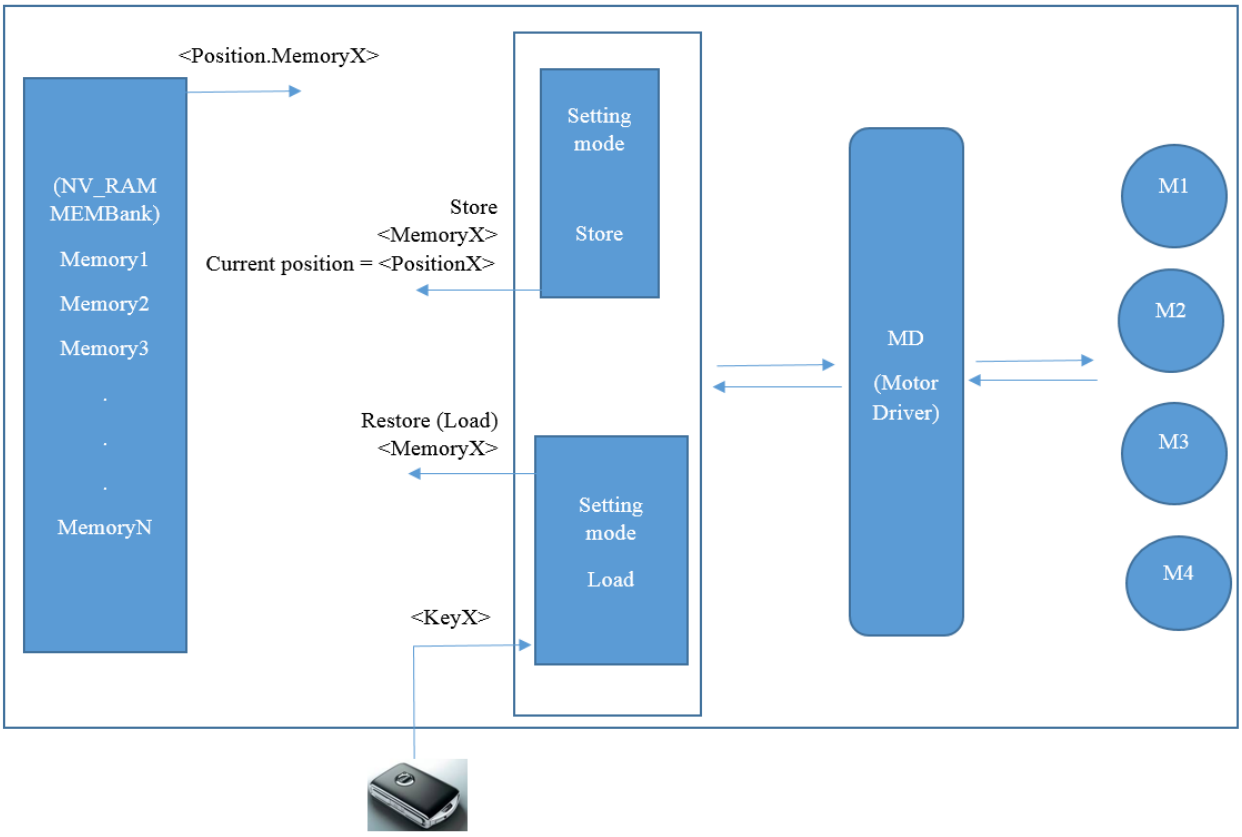

Figure 12. Recalling and storing seat position

5.4.2 User profile identification handling

The stored position can get allocated to the Profile IDs. The profile IDs can be activated when the user unlock the car using key no1-13. Each potential key can get delegated to a profile ID. According to this logic the active profile is the current profile and the requested adjustments is automatically set just after the driver entrance.

Hex value: 0x00 – 0x0F ⟺ User ID: 1-13

After unlocking, the transition between states gets triggered and also seat position adjustment starts.

5.5. SCM as Slave node and its transmission Structure

The setup studied in this thesis is a simple CAN master-LIN slave system. As mentioned, Power Seat Module is the CAN node and the master ECU for the Seat Comfort Module (SCM). In a master-slave LIN communication, SCM receives the configurable frame that are defined on LIN Description File. Data is transmitted and received between two nodes with fix structure. Data-frame has a break field to show an incoming frame. Data filed has the transmission information which is supported by a checksum and counter. Data-path is sent by both of the nodes. Transmit frames are sent from the PSM to the SCM (master to slave) and response frames in opposite direction, where the header is always coming from the PSM. Messages are communicated on a specific tic-time which depends on the LIN speed (version) and message length. A LIN node can also be in control of another LIN slave. In this setup the number of LIN slaves are limited to 4 or 5 ECUs. Figure 13 shows the frame structure transmitted between the PSM and the SCM node. As can be seen, there are unconditional frames containing application level signaling and transmitted during default session. Diagnostic frames are transmitted on the programming session.

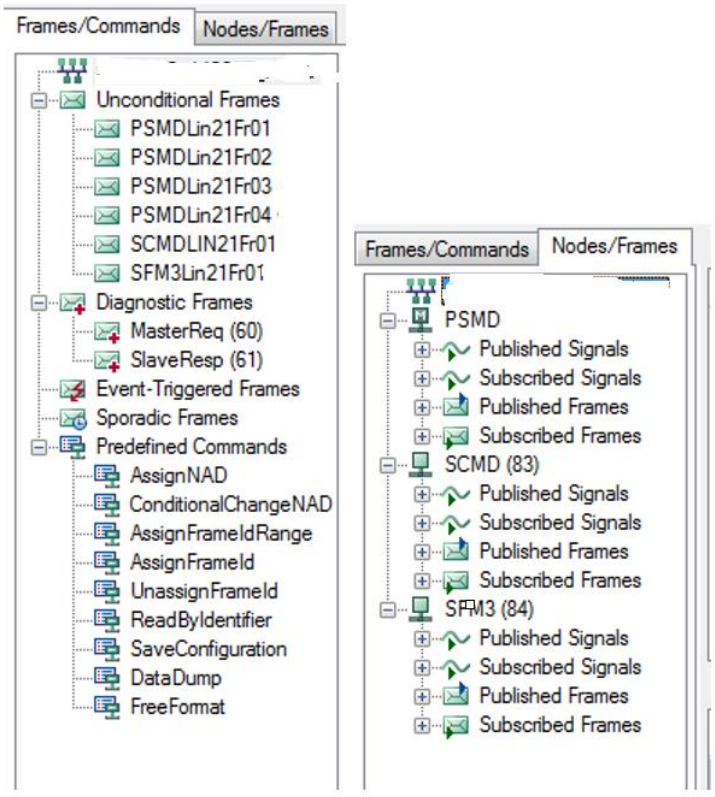

Figure 13. Unconditional frames, diagnostic frames and event triggered frames.

5.6. MATLAB shell model for PSM module

In order to build a shell model for the application layer the following subsystems and blocks are prepared. Block 1-4 are the main simulation blocks taking the input values from the switch pack and other nodes on the network. In order to keep the modularity and also to have an error-free logic, each block is supported by a subsystem (sub-blocks).

Following is brief description for the blocks and the subsystem:

1. **Power Seat Module driver**

    a. **Switch interpreter**

        i. Switch status filtered decoder: Since switch input is coded, it needs to get decoded using this block.

    b. **Functional Mode controller**

        Functional mode controller: Handles toggling between different modes using the anti-pinch flag and the profile positions and the mode management signals as input. The sensors values and positions are used in order to control the transition between different modes upon the request signals. These values return the functional mode as block output.

        i. **Motor Controller :**Handles the motor priority during movement , using the blocks of actuator groups which is fed by the movement offset (the difference between actual and requested position)
            1. **Seat actuator Control output for angle 1:** If switches are pressed to move actuator, and actual movement flag is on Idle, then send control signal to actuator 1.

            2. **Seat actuator Control output for angle 2:** If switches are pressed to move actuator, and actual movement flag is on Idle, then send control signal to actuator 2.

            3. **Seat actuator Control output for angle N:** If switches are pressed to move actuator, and actual movement flag is on Idle, then send control signal to actuator N.

    c. **Seat axis controller:** This is the block is to evaluate if the movement is possible to perform or not. This matter is done using the movement command signals. If this set of signals allows the movement then the switch signal (from switchback or multifunction) decides which motor (motors) to be activated

i. **Pause on crank:** This block changes the seat movement command signal and deactivates the movement, under the situation that seat movement is not desirable such as engine crank.

ii. **Preset position control:** To decide if the auto-movement is valid to start. Also to decide when to start the auto movement and when the movement is finished and the flag can be set to idle.

iii. **Multifunction switch Control:** To decide the lumbar and headrest position based on multifunction switch controller input and also the specified usage mode (if the movement is allowed).

iv. **Signal conversion:** Takes the input of multifunction switch which is in enum type and decode it and convert it to the driver seat enum type with idle as the default value.

d. Memory bank: As mentioned before on profile personalization chapter, based on key profiles that unlocks the car door, a personalized profiled gets loaded and preset adjustment is restored. This block simulates the functionality by the active-profile using a flag that shows what to read or write on active profile.

i. Write to memory: Using the active profile as input, this block decides to store a position on NV-RAM based on the requested position.

ii. **Read from memory:** Returns the requested position using active profile information.

e. User profile id determination: To check the range of user profile id, it shall not exceed the maximum number of memories.

2. **PSMD seat and actuator plant model**: This plant model is necessary to increment the position value if conditions are met. Positions can be incremented based on the request on each actuator.

3. **Power Seat Module passenger:** In case of having the PSMP (passenger seat module), driver module shall control the passenger node. PSMP implements the same functionalities as PSMD except differences in memory handlings and automatic adjustments.

4. **PSM Passenger seat and actuator plant model**: Handles the position incrimination in passenger module on the same way as driver module.

5.7. PSMD-SCMD set up

This part it to elaborate the setup to build a system that runs with simulated PSMD and physical SCMD that comes with air-bladders pumps and actuators. In this setup SCMD is connected to the LIN network as a slave to the CAN master, Power Seat Module. Signaling between SCMD and PSMD is defined on LDF (LIN description files) files dedicated for the respective LIN network. Figure 14 shows the general setup and also the connection between the developed CANoe panel and the MATLAB shell model.

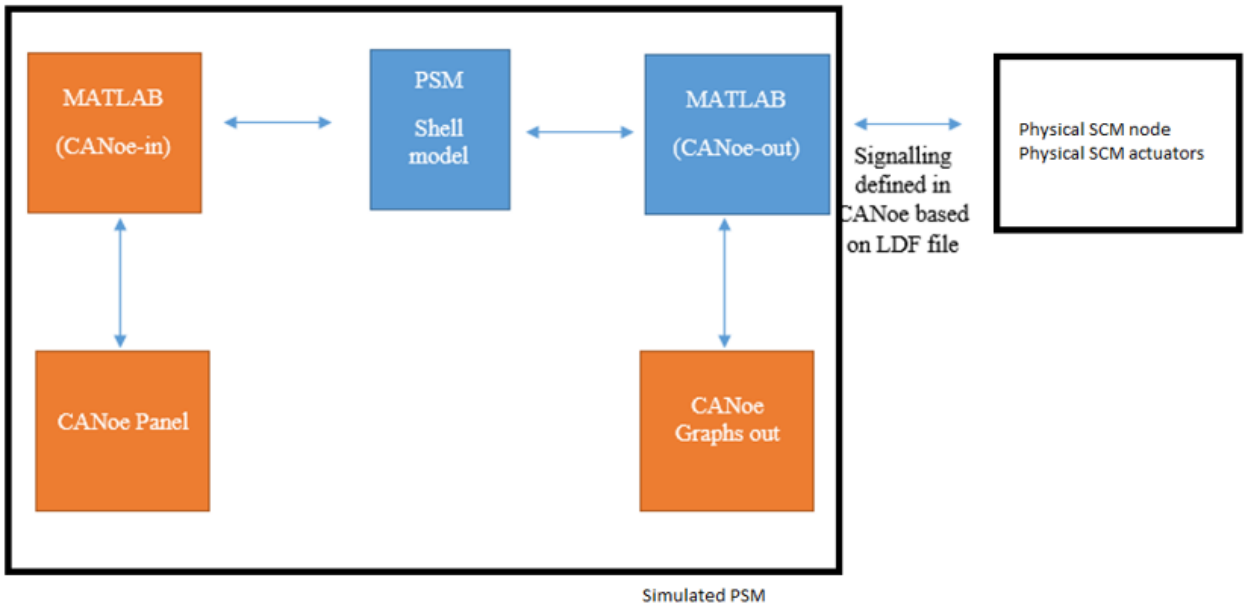

Figure 14. suggested setup , where the orange colored blocks are in CANoe and Blue colors in MATLAB Simulink

5.8. CANoe-MATLAB integration:

In CANoe there are different application areas these areas are analysis, simulation, test, diagnosis. MATLAB-CANoe supports the areas of analysis and simulation. In order to integrate the Simulink model in CANoe simulation environment, C-code shall be generated from shell models in CANoe. Microsoft visual studio is needed in order to generate the C code and the C code can be generated from MATLAB version of 2007a or newer. As the layer structure of Simulink node described on previous chapter, the exchange between application and interaction layer, is already facilitated by singling. Therefore, no network information needs to get defined in application layer. CANoe-MATLAB interface (Figure 15) installs a block-set in MATLAB Simulink blocks. This block-set contains the input-output, the signals and the system variables. Using this block-set, the subsystems in MATLAB can get triggered and also CAPL (CAN Access Programming Language) programs can be called.

Figure 15. Vector CANoe block-set in SIMULINK library.

5.9. Simulation mode

Figure 16 shows how to switch between the offline and the synchronized mode in CANoe. Simulation mode is to define how CANoe and MATLAB get synced. Simulation mode can be set and adjusted in CANoe and has three different modes:

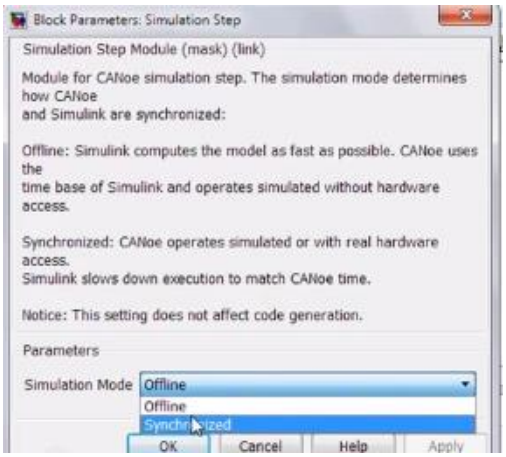

Figure 16. Simulation mode setup in CANoe

1. Offline mode: There is no access to physical hardware in offline mode. The system configuration is completely simulated.

2. Synchronized mode: The Physical hardware can be operated in synchronize mode. Simulation is done in real-time in this mode.

3. HIL mode (hardware in the loop mode):

Figure 17 shows how to generate the C code from the shell model using CANoe as the target in the HIL mode. Parameterization gets activated to let the parameters get tuned in real-time.

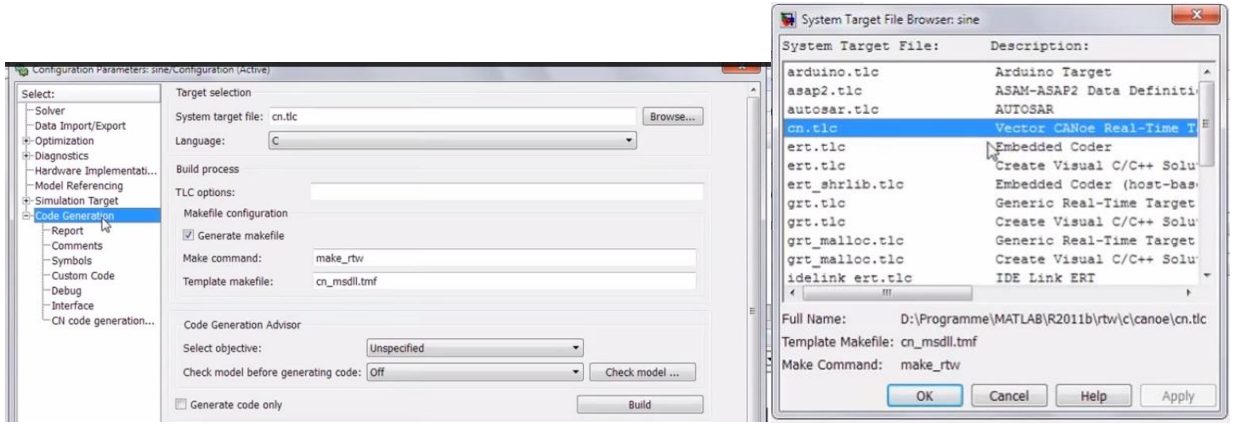

Figure 17. Select C code and use CANoe as target.

5.10. Link the shell models to CANoe

Figure 18 shows how to assign the generated code. In order to assign the DLL file to the simulation node in CANoe where operator integrate the model file and other generated files.

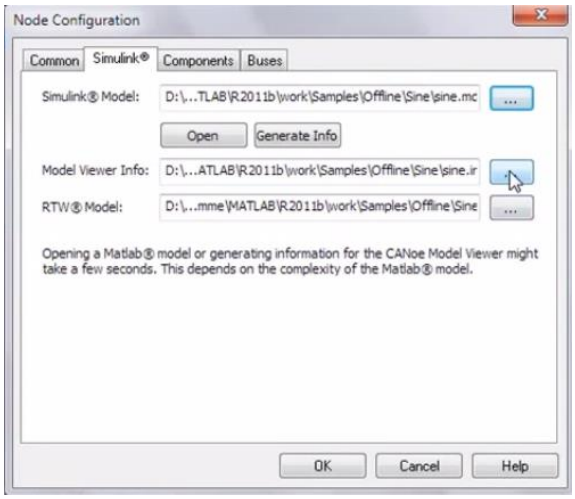

Figure 18. Link the code to simulated node

After measurement start, the target component is operated in real-time. An advantage of the HIL mode is that the system parameters can get adjusted during component operation in real-time and also that models or state flow charts can be displayed accordingly.

5.11. DUT CANoe panel

Using CANoe, user can assign a symbol to adjust the system signal values without any panel or CAPL (CAN Access Programming Language) program. This can be done in symbol panel from the shortcut menu. Panel automatically configures and receives the symbol selected values. To simulate the whole process and

functionality user shall build a test-panel or script that calls the series of symbols successively. Panel designer is been used for this matter. This is a powerful tool in CANoe in order to tune the values and also on-demand adjustment in parameters and bus signals. CANoe panel can be adjusted as a source to send the data to the other controllers, and also to observe, log and analyze the traffic on the CAN/LIN bus. In order to drive the physical SCM node from simulated PSM, the following CANoe panel shown in Figure 19 is prepared. It contains the basic seat positions that are called through symbols from the graphical interface. Symbols in the panel are linked to the CAN bus values to configure the signal data-elements on simulated PSM. These symbols can also send commands to imitate the massage pumps on physical SCM node.

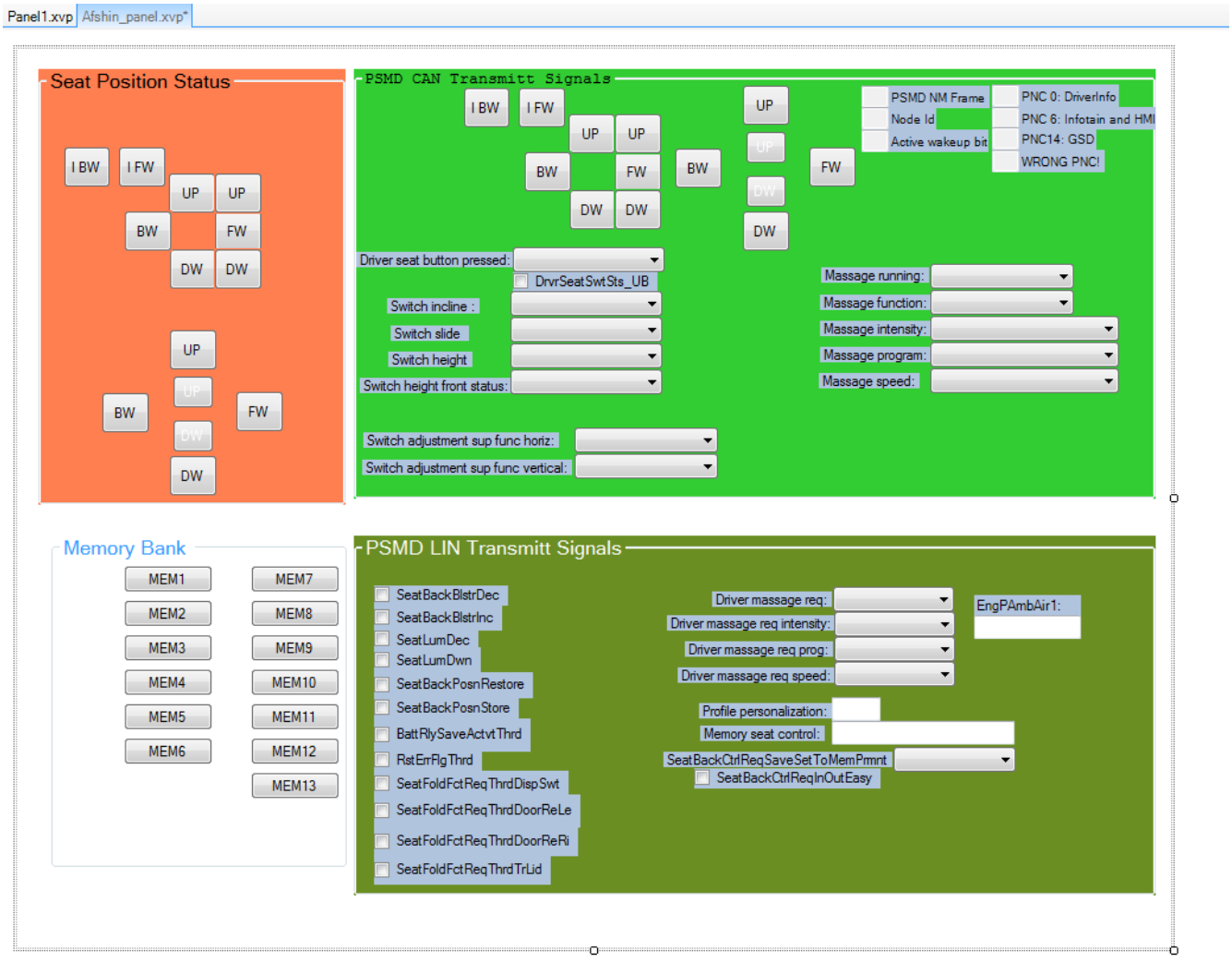

Figure 19 PSM CANoe panel

As figure 19 shows, the memory bank buttons request one of the 13 predefined saved position. These buttons send the command signal to the simulated PSM and SCM as the slave node can react upon the request.

6. Verification method

Following is the test cases to verify the physical SCM's functionality using the CANoe panel connected to simulate the PSM. Test case steps describe how to operate the test scenario. Test description gives idea about the aim and the target of the test. The test result can be seen at the end of the test cases, and it contains the observed result after performing the test case.

TEST-CASE1, TEST-CASE5.manual adjustment of bolster
Expected result: Back bolster shall start moving.

Test steps 1- Start CANoe, CAN/LIN 2- Usage mode => active 3- Power supply SCM 4- On CANoe panel : use adjustment button up-down-in-out for SCM bolster
Test description: Sending the bolster movement command using CANoe panel and through the simulated PSM Test result: Physical SCM receives the commands and starts changing pressure on the bolsters.

TEST-CASE2, Memory bank
Expected result: Back bolster pressure shall change

Test steps:

1- Start CANoe, CAN/LIN
2- Usage mode => active
3- Power supply SCM
4- On CANoe panel : Profile personalization signal is set between 1-13
   Note :
   If Memseatcntrl=0 -> use the membank1-13 (profile id)
   If Memseatcntrl=1-3 -> use personal setting (door switches memory bank 1-3)
5- On CANoe panel :Seatbackposnstor=store
   Note :Store vs restore (Boolean signal) to show is we want to write or read (in out easy = false)
6- Set the position status on Power Seat Module on CANoe penal.
7- set the request signal on CANoe panel
8- PSM sends the request to physical SCM
9- Physical SCM bolster is adjusted according to memsearctrl no X

Test description:

After setting the status signals of the seats. Request signal is sent by Power Seat Module to SCM.
P.S. it's a "simulated" PSM that sends the request to the "real" SCM.

Test result:

The predefined lumbar/bolster/motor positions are stored in memory no #1 and adjusted accordingly on physical Seat Comfort Module. Bolsters are adjusted to predefined pressure and the pressure change is observable.

TEST-CASE3, EmgyStop

Expected result: back bolster adjustment shall stop immediately after emergency stop

Test steps: 1- Start CANoe, CAN/LIN 2- Usage mode => active 3 Emergency stop signal on CANoe panel is set as true. 4 its shall freeze the bolster on physical SCM 5 as long as the Emergency stop signal is true, the bolster shall not activate.
Test description: Emergency stop function is to prevent sending the double commands to a single motor. When higher priority command has sent to either PSM or SCM the current movement shall be stopped immediately Test result : After PSM sends the position command signals to SCM, all the back bolster movement on physical SCM is immediately frozen. After the emgstop signal changes value to dative the previous bolster movement continue to reach the requested pressure.

TEST-CASE4, TEST-CASE5.Easy Ingress: In-out-easy signal

Expected result: Back bolster pressure shall change to zero in order to make easy entrance for user

Test steps: 1- Start CANoe, CAN/LIN 2- Usage mode => active 3- Power supply SCM 4- Profile personalization signal is set between 1-13 On Canoe panel : Memseatcntrl=1-3 -> use personal setting (door switches memory bank 1-3) 5- On CANoe panel Seatbackposnstor=store (Boolean signal) to show is we want to write or read (in out easy = false). 6- on CANoe panel : set in-out-easy button = true 7- SCM releases all the air from bolster and air cells.
Test description: Door opening status signal shall sent to PSM by CANoe panel, after receiving this signal PSM and SCM shall reaches the minimum position/pressure so the driver can get in the car comfortingly . Test result: Easy entry signal is chosen to positive in the panel. Simulated PSM has modified its position to minimum value of the motion range and also physical SCM has deflated its bolster to minimize the pressure.

7. Conclusion

Verification of a LIN slaves using the simulated master node is a good solution, as physical verification method using both master and slave node is impossible to perform because of unavailability of physical master node.

The proposed solution is a good alternative in case:

1. Master nodes is not physically available
2. The logical solution for master node is available since the beginning of development phase and just after the concept phase.

The suggested solution gives independence to start verification and not waiting for the master node to reach the same SW maturity level as slave node. Using the test feedback in earlier development phase, system constructors can modify the design to build a reliable system. In reality, simulated module can present part of systematic problems and mismatches between functionalities of master and slave node using physical nodes. Not all systematic and function issues can be simulated through CANoe environment.

Simulation based verification is helpful to find functional problems when either of nodes is not present to build a physical test setup. Fault cases such as switch stuck, cable problems, overheating and anti-trap system (anti pinch) failures and all mechanical limitations can only be analyzed using physical hardware. Tunings on motor range of motion need calibration of whole system in presence of the physical actuators. Simulation based verification cannot be the only testing strategy and shall ideally be the backup plan. It shall start on the initial phases of development. Feedbacks from simulation based verification shall be used to correct the system design as early as possible in order to avoid drastic expenses on second wing of V-shape development strategy.

8. Future research

As discussed in previous section, it would be interesting to further analyse the overheating effect, and to study how the anti-trap system failure could affect the system functionality as a whole, when using the simulated master node and physical slave.
As it described on verification section, latencies between the expected manoeuvring time and the physical manoeuvring time result in further malfunction of the whole system and can lead to hardware damage. The question is that if approximate operation period using the measure component parameters can get simultaneously corrected, and get adjusted automatically?

This thesis is focused entirely on shortening the verification time when slave node can't get verified as a stand-alone unit. It would be interesting to see how both simulated components can verify the physical system and even how to minimize the simulation time. Since the time to complete the Simulink model can even exceed the development time of physical hardware.

References

[1] H. K. F. W. a. R. E. K. Tindell, Safe automotive software development, Washington, DC, USA : IEEE, 2003.

[2] L. Consortium, "LIN Specification Package Revision 2.1," November 24, 2006.

[3] V. G. Mircea Popa and A. Botas, Lin Bus Testing Software, Automation and Computers, "Politehnica" University of Timisoara, ROMANIA: IEEE.

[4] A. R. &. E. Wallin, "LIN – Local Interconnect Network – for use as sub-bus in Volvo trucks," CHALMERS UNIVERSITY OF TECHNOLOGY, Göteborg, 2003.

[5] A. Deuter, "Slicing the V-Model -- Reduced Effort, Higher Flexibility," *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on,* pp. 1 - 10, 26-29 Aug. 2013.

[6] John O. Clark, "System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective," *IEEE SysCon 2009 —3rd Annual IEEE International Systems Conference,,* 2009.

[7] J. J. Johan Elgered, "AUTOSAR Communication Stack Implementation," Chalmers University of Technology, Gothenburg, March 2012.

[8] G. E. Andreas Deuter, Measuring the Software Size of Sliced V-Model Projects, PHOENIX CONTACT Electron. GmbH, Bad Pyrmont, Germany: IEEE.

[9] S. REY, Introduction to LIN - (Local Interconnect Network), digitales, May 13 2003.

[10] C. L. Renjun Li and F. Luo, A design for automotive CAN bus monitoring system, College of Automotive Engineering, Tongji University, Shanghai, China: IEEE.

[11] S. Corrigan, "Introduction to the Controller Area Network (CAN)," Texas Instruments Incorporated, July 2008.

[12] S. C, Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger, Chicago, Illinois : Illinois Institute of Technology .

[13] S. Lorenz, "The FlexRay Electrical Physical Layer Evolution," in *AUTOMOTIVE FLEXRAY*, 2010.

[14] AUTOSAR, "AUTOSAR Publications for Release 4.2- available at http://www.autosar.org/," 2014.

[15] B. X. Lihong Jiang and H. Cai, A multi-views modeling approach for product lifecycle management in supply chain, Sch. of Software, Shanghai JiaoTong Univ., Shanghai, China: IEEE.

[16] M. Deloge, A highly-digitized automotive CAN transceiver in 0.14m high-voltage SOI CMOS, NXP Semicond., BU Automotive, Eindhoven, Netherlands: IEEE.