International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.2, Issue No.7, pp : 733-736 1 July 2013

AUTOSAR for Local Interconnect Network Management in Basic Software Modules

Tejaswini. B. L[1],Dr.A.S.Poornima [2] & Dhruti S[3]

[1,2] Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka, India.

[3] Project Leader, KPIT Cummins Infosystem Ltd., Bangalore, Karanataka, India.

tejaswini.bl12@gmail.com, aspoornima@sit.ac.in, Dhruthi.s@kpitcummins.com

Abstract-The AUTOSAR(Automotive Open System Architecture) is open and standardized automotive software architecture. Developed by automobile manufacturers, suppliers and tool developers. Its is an alliance of OEM manufacturers and Tier 1 automotive suppliers working together to develop and establish a de-facto open industry standard for automotive E/E architecture which will serve as a basic infrastructure for the management of functions within both future applications and standard software modules. AUTOSAR supports the re-use of software and hardware components of automotive electronic systems. In this paper gives Autosar layered architecture, and LINNM overview. Major LIN applications are some integration equipment such as, Door's, steerin wheel, seats and air-conditoning. The AUTOSAR LinNm Module is the hardware independent protocol that can only be used on LIN. The LinNm function provides an adaptation between Generic Network Management Interface (Nm) and LIN Interface (LinIf) module. Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network.

Keywords
Local interface network management(LINNM), Basic software module(BSW), AUTOSAR(AUTomotive Open System Architecture).

I. INTRODUCTION

The AUTOSAR is open and standardized automotive software architecture. Developed by automobile manufacturers, suppliers and tool developers. Enables the use of a component based software design model for the design of a vehicular system. In the near future, today's hardware and component-driven development process will be more and more replaced by a requirement and function-driven process. Future engineering does not aim at optimizing single components but optimizing on system level. This requires an open architecture as well as scalable and exchangeable software modules. Since these issues can hardly be handled by individual companies and constitute an industry-wide challenge, leading OEMs and Tier 1 suppliers have jointly decided to establish an open standard for automotive E/E architecture, leading to the AUTOSAR development cooperation. Basic idea is the re-use of software components so that the rising complexity can also be handled in the future. The standardized aim of autosar is scalability, transferability, better integration and maintainability.

The goals of AUTOSAR[1] are:

- Fulfillment of future vehicle requirements, such as, availability and safety, software upgrades/ updates and maintainability.
- Increased scalability and flexibility to integrate and transfer functions.
- Higher penetration of "Commercial off the Shelf" software and hardware components across product lines.
- Improved containment of product and process complexity and risk.
- Cost optimization of scalable systems.

II. AUTOSAR LAYERED ARCHITECTURE

To make a component based design possible, the AUTOSAR project uses a layered architecture that ensures the decoupling of the functionality from the supporting hardware and software services. The layered architecture is used on every ECU(Electronic Control Unit) and makes it possible to design a vehicle system without thinking in terms of ECUs. The designers select a number of software components that do not know on which ECU certain software components are installed or hardware is connected.

Microcontroller Abstraction Layer

The Microcontroller Abstraction Layer (MCAL) is located at the very bottom of the BSW layer. MCAL uses its internal software drivers to directly communicate with the microcontroller. These drivers include: memory-, communication- and I/O drivers. The task of the layer is to make layers above it microcontroller independent. When the MCAL is implemented it is microcontroller dependent but provides a standardized- and microcontroller independent interface upwards in the stack thus fulfilling its purpose.

International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.2, Issue No.7, pp : 733-736 1 July 2013

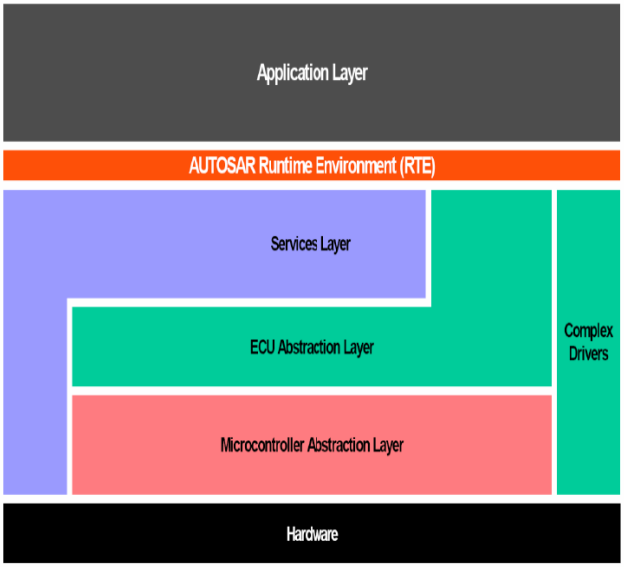


Figure1: Layered Architecture

ECU Abstraction Layer
Located on top of the MCAL is the ECU Abstraction Layer (ECUAL). Internally it has drivers for external devices and uses the interface defined by the MCAL to access drivers at the lowest level

Complex Drivers Layer
The Complex Drivers Layer (CDL) is one of two layers spanning the entire BSW layer. Typically this layer is used to integrate special purpose functionality and functionality currently being migrated from a previous system. Since this is the only layer between the microcontroller and the RTE, drivers for devices with strict timing constraints can benefit from being placed in the CDL as the multi-layered parts of the BSW layer is likely to introduce overhead due to additional layers and standardization .

Runtime Environment
The RTE is the layer between the application layer and the BSW layer. It provides the application software with services from the service layer. All communication between SWCs, either on the same ECU or different ones, or services are done via the RTE. The main task of the RTE is to make the layer above and below it completely independent of each other. In other words, SWCs running on an ECU have no idea what the ECU looks like hence a SWC will be able to run on different looking ECUs without any modifications.

Logically the RTE can be seen as two sub-parts realizing different software functionality :

1. Communication
2. Scheduling

When the RTE is implemented it is ECU and application dependent thus it is specifically generated for each different looking ECU. Instead of adapting SWCs to different ECUs that is what is done with the RTE, this way SWCs can stay the same thus accomplishing the set out task.

Application Layer
The Application Layer differs in two major aspects from the rest of the layers, it is component based and not standardized. SWCs are such a key part of this layer and AUTOSAR and have as a result. Looking at the RTE and the BSW layers there are lengthy standards documents specifying how they shall look and behave. Creating SWCs and the behavior in the application layer can be done freely according to what a particular vendor wants. One restriction though is that all communication with other components, whether it is intra- or inter-communication, has to be achieved in a standardized way by using the RTE. Since the RTE makes layers above it independent of hardware such as the ECU and the microcontroller the application layer is apart from a few cases, only dependent on the RTE. For example the sensor-actuator SWC is dependent on the hardware but SWCs communicating with it are not.

III. ECU SPECTRUM

ECU Spectrum takes ECU Configuration Parameter Definition File(s) as input at the time of the creation of new project. This ECU Configuration Parameter Definition is in XML format and contains definition for Modules, Containers and Parameters. The format of the XML file must follow the AUTOSAR ECU specification standards.

Module: Modules denote an ECU Configuration Parameters Software Module. Individual modules are assigned a unique name. Number of module instances depends on multiplicity of the module.

Container: Containers are used to group parameters and references. The number of instances of a container depends on multiplicity of the container. A Sub-Container is also used to group parameters and references. Sub-container is a part of container. Sub-containers are defined in containers. The number of instances of a container depends on multiplicity of the container.

Multiplicity: Multiplicity is used to specify how often a specific configuration element (module, container, parameter or reference) may occur in an ECU Configuration Description file. Lower-Multiplicity and Upper-Multiplicity are two attributes to specify minimum and maximum occurrences. In any case Lower-Multiplicity should be less than or equal to Upper-Multiplicity.

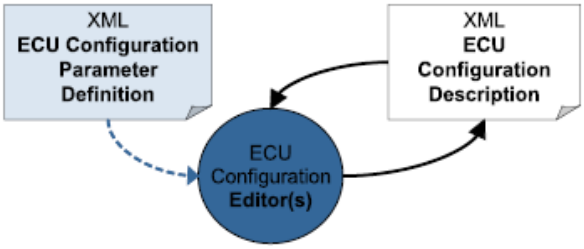
Lower-Multiplicity is mentioned as 1 means that the element is mandatory. Lower-Multiplicity mentioned as 0, means that the

element is an option. Upper-Multiplicity mentioned as * means that the parameter can occur any number of times.

ECU configuration Parameter Definition File contains container definition, parameter definition and reference definition.

- Container definition: Group of parameter, reference and sub-containers.
- Common attributes: applicable for parameter and reference with short name, description, lower and upper multiplicity.
- Additional attribute: Multiple configuration container and post build changeable.

Based on the implementation, lower and upper multiplicity can be modified with in the autosar defined range.



Parameter definition: Provide set of control for configuration of the parameter and common attribute implements configuration class, origin and symbolic name value and additional attributes are default value as optional and some parameter its mandatory.

IV. SYSTEM OVERVIEW OF AUTOSAR LINNM ARCHITECTURE

The Local Interrconnect Network(LIN) is a new low cost serial communication system. The communication is based on the SCI data format, a single –master/multiple- slave concept bus and clock synchronization for nodes without a stabilized time base. The AUTOSAR LinNm Module is the hardware independent protocol that can only be used on LIN. The LinNm function provides an adaptation between Generic Network Management Interface (Nm) and LIN Interface (LinIf) module. Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network. It takes the description file as input which is generated by the output of ECU spectrum. It's a part of communication services layer in AUTOSAR environment and is as shown in the figure below:

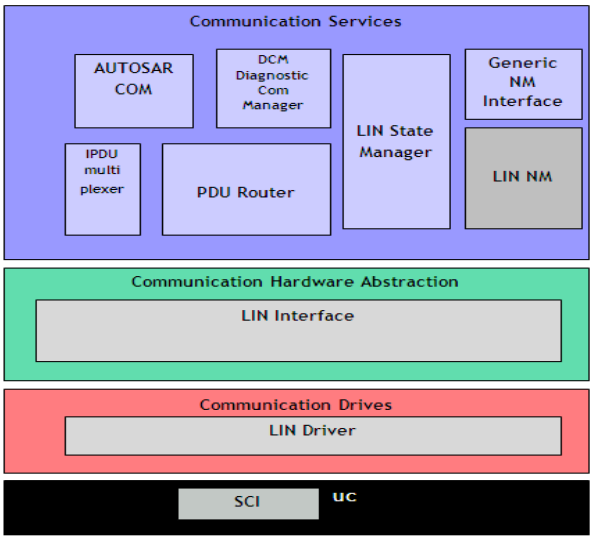


Figure2: System Overview of AUTOSAR LinNm Architecture.

LIN bus is used in some integration equipment, such as doors, streeing wheel, seats and air- conditioning. LIN enables a cost effective communication for smart sensor and actuators. Its using digital interface encoding instead of analog interface encoding, electronic devices can be easily connected to in- vehicle network system and implementation for different diagnosis and maintenance function system.

The LIN-Bus (Local Interconnect Network) is a computer networking bus-system used within current automotive network architectures. LIN specification is enforced by the LIN-consortium.

The main properties of the LIN bus are [3]:

- single master with multiple slaves(up to 16) concept
- low cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software or as pure state machine.
- self-synchronization without a quartz or ceramics resonator in the slave nodes
- deterministic signal transmission with signal propagation time computable in advance
- low cost single-wire implementation
- speed up to 20 kbit/s.
- signal based application interaction
- predictable behavior
- re-configurability
- transport layer and diagnostic support

The LINNM bus operation is based on single-master/ multiple –slave concept shown in below figure: Each LINNM node is divided into two individual parts:

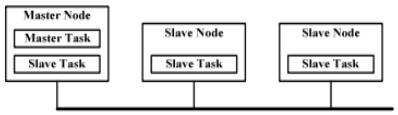


Figure3 : LINNM bus topology

Master Task, which decides the frame sequence using a schedule table. Slave Task, which is responsible for data transfer on the bus, and allow waking up the entire slave node from sleep state.

The LINNM configuration language description describes the format of the LIN configuration file, which is used to configure the complete network and serves as a common interface between the OEM and suppliers of the different network nodes as well as input to the development and analysis tools.

The LinNm Module is divided into following sub-modules based on the functionality required:

- Initialization
- Transmission control
- Information services
- Network Controlback Functions

Initialization

This sub module is responsible for initialization of the complete AUTOSAR LinNm Module, i.e. all channels that are activated at configuration time are initialized by providing LinNm_Init function. The init function is used initialize the module.

Transmission Control

This sub module is responsible for control of transmission of NM messages in the network. In this sub module all the services such as LinNm_EnableCommunication, LinNm_DisableCommunication, LinNm_RequestBusSynchronization and LinNm_Transmit are implemented to transfer controls.

When the communication needs between ECUs or it needs to switch to perform task then it enable communication by calling function, after completing task it will go to sleep mode, so save power or energy that disable communication function. The RequestBusSynchronization function is used to synchronize the bus request to perform one after the other.

Information Services

This sub module provides the functionality to 'get' or 'set' the LinNm related data, i.e. user data, node identifier and local node identifier of the LinNm Module and setting the repeat message bit of NM(Network Management) PDU(Protocol data unit) data. In this sub module all the services are implemented as empty functions.

Network Control

This sub module provides the functionality to change the Network state to "requested" or "released". It also provides the functionality of triggering the transition from Bus-Sleep mode to Network mode and Network mode to Bus-Sleep mode along with providing the information about the current state and current mode of the network management.

Callback Functions

This sub module consists of the call back services called by the LinIf module and implemented by the LinNm module. It provides the functionality to confirm the successfully processed LIN transmit request. The LINNM API describes the interface between the network and the application program. This concept allows the implementation of a seamless chain of design and development tools and enhances the speed of development and the reliability of the network.

V. CONCLUSION

The AUTOSAR LinNm Module is the hardware independent protocol that can only be used on LIN. The LinNm function provides an adaptation between Generic Network Management Interface (Nm) and LIN Interface (LinIf) module. Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network. The LinNm is used to save the power and energy level life. The mainly used in door locking system, steering wheel and seat adjustment system in vehicles.

References

[1] AUTOSAR. AUTOSAR partnershiphomepage: www.autosar.org.
[2] AUTOSAR. Layered Software Architecture.
[3] AUTOSAR: Specification of LINNMV2.0.1.
[4] AUTOSAR: Specification of AUTOSAR R4.0
[5]AUTOSAR Administration, "AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf", Document Id No. 053, V3.2.0, R4.0 Rev3, 06 October, 2011.
[6] AUTOSAR Administration, "Specification of LIN Interface (AUTOSAR_SWS_LinIf.pdf)", Document Id No. 073, V4.0.0, R4.0 Rev3, 28 October, 2011.
[7] LIN Consortium, "LIN Specification Package Revision 2.1 (LIN_Spec_Pac2_1.pdf)", 24 November, 2006
[8] AUTOSAR Administration, "Specification of LIN Driver (AUTOSAR_SWS_LINDriver.pdf)", Document Id No. 072, V1.5.0, R4.0 Rev3, 28 October, 2011.
[9] AUTOSAR Administration, "Specification of LIN Transceiver Driver (AUTOSAR_SWS_LINTransceiverDriver.pdf)", Document Id No. 257, V1.2.0, R4.0 Rev3, 09 December, 2011.
[10] AUTOSAR Administration, "Specification of LIN State Manager (AUTOSAR_SWS_LIN_StateManager.pdf)", Document Id No. 255, V1.3.0, R4.0 Rev3, 02 November, 2011