1. **What is AUTOSAR Communication Stack (ComStack)?**

## Definition of AUTOSAR Communication Stack (ComStack):

As shown in the diagram below, AUTOSAR has a layered architecture. The AUTOSAR development process is implemented across the following software layers (bottom-to-top):

* MCAL (Microcontroller Abstraction Layer)
* Basic Software Layer (BSW)
* Run-Time Environment (RTE)
* Application Layer

In this AUTOSAR layered architecture, Communication Stack or ComStack facilitates communication. Hence ComStack can be defined as a software stack that provides communication services to the Basic Software Modules and Application Layer or Application Software.

Learn more details about the AUTOSAR ComStack (Communication Stack)

Below is the depiction of a generic AUTOSAR Communication Stack

*(Source – autosar.org )*

Depending on the Bus Type of the in-vehicle network (such as CAN, LIN, Flex-Ray, MOST), implementation of the communication stack is executed. For example, if the underlying Bus type of the in-vehicle network is CAN, then CAN implementation of the communication stack is executed.

We will first focus on generic ComStack and then introduce you to the specification of CAN Communication Stack in AUTOSAR.

A generic Communication Stack in AUTOSAR layered architecture is a set of following software modules:

* AUTOSAR COM – part of the Services Layer
* Bus Specific Interface Modules – part of the ECU Abstraction Layer (For example -CanIf, LinIf, FrIf)
* External Bus Drivers – part of the ECU Abstraction Layer (For example – External drivers likeCanDrv, LinDrv, FlexrayDrv)
* Internal Bus Drivers – part of the AUTOSAR MCAL (For example – CanDrv, LinDrv, FrDrv)

Introduction to CAN Communication Stack in AUTOSAR :

The following diagram depicts the CAN based Communication Stack (ComStack):

List of different modules of the CAN Comunication Stack ComStack:

* COM(Services Layer)
* PDU Router(Services layer)
* CAN State Manager(Services Layer)
* CAN Network Manager(Services Layer)
* CAN Transport Protocol(Services Layer)
* CAN Interface(ECU Abstraction Layer)
* External CAN Driver(ECU Abstraction Layer)
* CAN Driver(MCAL Layer)

*(Source – autosar.org)*

Some of the modules of the Communication Stack perform functions related to vehicle diagnostics, PDU multiplexing and CAN Transceiver.

Learn more details about the CAN ComStack software modules:

CAN Stack BSW Modules:

* + AUTOSAR COM: AUTOSAR COM is a module between the RTE and the PDU Router. It is responsible for providing Signal level access to the application layer and PDU level access to the lower layers independent of the protocol.

It packs the signals to a PDU at the transmitter and unpacks the received PDU to provide signal level access to the application at the receiver. At the PDU level, COM is responsible for grouping of the PDUs, starting and stopping of the PDU groups.

* + PDU Router: PDU Router is a module responsible for routing the PDU to the respective Bus Specific Interface modules.

Above the PduR module all the PDUs are protocol independent, and below PduR all the PDUs are routed to the protocol specific modules. PduR is also responsible for PDU level gatewaying i.e. transmitting the received PDU from one Bus Specific Interface module to other Bus Specific Interface module.

Gatewaying can also be done when a PDU is to be routed from one controller to another over the same protocol.

* + Can TP: The basic services offered by the Can TP module are segmentation of messages which have a payload of more than 8 bytes, transmission of the messages with flow control and reassembling the segmented messages at the receiver.
  + Can Interface: CAN Interface(CanIf) is a module in the ECU Abstraction Layer which is responsible for services like Transmit Request, Transmit Confirmation, Reception Indication, Controller mode control and PDU mode control.
  + Can State Manager (CanSM): This module shall implement the control flow for the respective bus. The CAN State Manager (CanSM) is a member of the Communication Service Layer.

It interacts with the Communication Hardware Abstraction Layer and the System Service Layer.

* + CanNm: The AUTOSAR CAN Network Management is a hardware independent protocol tools that can only be used on CAN network.

Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network. The CAN Network Management (CanNm) function provides an adaptation between network Management Interface (NmIf) and CAN Interface (CanIf) module.

* Can Driver (CanDrv): This module is a part of the MCAL layer and provides hardware access to the upper layer services and a hardware-independent interface to the upper layers. CanIf is the only module that can access the CAN driver.

**What is AUTOSAR MCAL? Learn about the software module architecture and device drivers**

This blog is part of our series on understanding AUTOSAR better. Our Automotive Product Engineering team, who has in-depth [AUTOSAR](http://www.embitel.com/product-engineering-2/automotive/autosar) expertise, will help us understand the essential components of the layered architecture of AUTOSAR.

To kick start our journey of learning more about AUTOSAR, in this blog we will first focus on AUTOSAR MCAL.

## Basics of AUTOSAR MCAL:

MCAL stands for Microcontroller Abstraction Layer. In the context of embedded software development, the MCAL can be defined as follows:

MCAL is a software module that has direct access to all the on-chip MCU peripheral modules and external devices, which are mapped to memory. And it makes the upper software layers (Basic software layer, or BSW, Application Layer) independent of the MCU.

MCAL enables a very significant advantage of the layered architecture of the AUTOSAR compliant design as it makes the application and also the middleware (Basic Software layer) independent of the underlying hardware platform.

This renders immense benefit to the product development cost and time, as there is a shift in the [ECU design approach](http://www.embitel.com/product-engineering2/embedded-casestudies/ecu-software-migration-for-a-cv-client) from coding to configuration.

### **What are the different device drivers of AUTOSAR MCAL module?**

Microcontroller Drivers:

* GPT Driver: GPT (General Purpose Timer) device driver uses on-chip MCU timer. Initializes GPT and performs timer count.
* WDG Driver: WDG (Watchdog) Driver, this on-chip device driver Initializes WDG and performs WDG mode settings.
* MCU Driver: MCU (Micro Controller Unit) Driver, this device driver helps configure MCU settings, initializes clock and helps configure power mode settings.

Memory Drivers

* FLS Driver: FLS (Flash) Driver initializes FLS and reads/writes to FLS memory.

Communication Drivers

* SPI Handler/Driver: SPI (Serial Peripheral Interface) is a Handler/Driver Device with on-chip clock serial function that Initializes SPI, performs SPI input/output and SPI I/O buffer settings
* LIN Driver: LIN (Local Interconnected network) is a device driver that initializes LIN and performs LIN input/output.
* CAN Driver: CAN (Controller Area Network) is a device driver that initializes CAN and performs CAN input/output.
* FlexRay Driver: FlexRaydevice driver initializes FlexRay and performs FlexRay input/output.
* Ethernet Driver: Ethernet device driver initializes Ethernet Driver and performs Ethernet Driver input/output.

I/O Drivers

* ICU Driver: ICU (Input Capture Unit) is a device driver using on-chip MCU timer and initializes ICU. It also measures PWM waveforms.
* PWM Driver: PWM (Pulse Width Modulation) is a device driver using on-chip MCU timer. It initializes PWM and sends PWM waveforms as output
* ADC Driver: ADC (Analog Digital Converter) is a device driver for on-chip ADC. It Initializes ADC, starts/stops AD conversion, sets AD conversion result buffer and reads AD conversion results.
* DIO Driver:DIO (Digital Input/Output) is an MCU port device driver thatperforms port signal (input/output).
* PORT Driver:PORT Driver is a MCU port device driver that performs MCU pin settings (I/O, shared functions)

**What is AUTOSAR Memory Stack | Software modules and device drivers**

This blog is part of a series of articles to help you learn about layered architecture of AUTOSAR.

Through this article our AUTOSAR development team, which is based in Bangalore, India, has tried to share knowledge with the community of AUTOSAR developers and automotive OEMS and Suppliers.

In this blog we will introduce you to AUTOSAR Memory Stack (MemStack). This will help you understand the software modules and device drivers associated with Memory Stack.

## Introduction: AUTOSAR Memory Stack (MemStack):

Memory Stack (MemStack) provides basic memory management services to the upper Application layer and to the Basic Software Modules (BSW) of the AUTOSAR layered architecture.

The memory management services ensure access to the memory cluster, to the devices or software functions, for reading and writing data to non-volatile memory media like Flash or EEPROM

The following block diagram show various software modules and device drivers associated with AUTOSAR Memory Stack:

### **Software Modules and Device Drivers – AUTOSAR MemStack**

Memory Stack in AUTOSAR layered architecture is a collection of software modules and device drivers.

Following is the list of modules in different layers of AUTOSAR:

* Non-Volatile Memory Manager (NvM) it is part of the AUTOSAR Services Layer
* Memory Interface (MemIf) it is part of the AUTOSAR ECU Abstraction Layer
* Flash EEPROM Emulation (Fee) it is part of the AUTOSAR ECU Abstraction Layer
* EEPROM Abstraction (Ea) it is part of the AUTOSAR ECU Abstraction Layer
* Flash Driver (Fls) it is part of the AUTOSAR MCAL Layer
* EEPROM Driver (Eep) it is part of the AUTOSAR MCAL Layer

### **Description of AUTOSAR Memory Stack software modules**

* Non-Volatile Memory Manager (NvM): The NvM module ensures data storage and maintenance of NV (non volatile) data according to the individual requirements in an automotive environment.

The NvM module manages the NV data of an EEPROM and/or a FLASH EEPROM emulation device.

* Memory Interface (MemIf) Module: The Memory Abstraction Interface (MemIf) module facilitates abstraction from the underlying FEE and EA modules. Hence MemIf module provides upper layer (NvM) with a virtual segmentation on a uniform linear address space.

This ensures that the Non-Volatile Memory Manager (NvM) is independent of the driver interface layers of EEPROM (Eep) and Flash interface (Fls)

* EEPROM Abstraction (Ea): EEPROM driver provides services for reading, writing, erasing data to/from an EEPROM. It also provides a service for comparing a data block in the EEPROM with a data block in the memory (e.g. RAM).

Ea module facilitates abstraction from the addressing scheme of underlying EEPROM driver and hence provides a uniform addressing scheme.

This ensures that the upper layer (NvM) need not be changed if the underlying EEPROM driver and device is replaced.

* Flash EEPROM Emulation (FEE) Module: The Flash EEPROM Emulation (FEE) abstracts from the device, a specific addressing scheme and segmentation.

This provides the upper layers (NvM) with a virtual addressing scheme, segmentation as well as a â€œvirtuallyâ€ unlimited number of erase cycles.

* Flash Driver (Fls): Fls Driver Initializes Flash and reads/writes to Flash memory.
* EEPROM driver (EeP): EEPROM driver provides services for reading, writing, erasing to/from an EEPROM.

It also provides a service for comparing a data block in the EEPROM with a data block in the memory (e.g. RAM).