Mohamed Abdelhay
April 2021

(**AUT**omotive **O**pen **S**ystem **AR**chitecture)

is a worldwide development partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry.

Course Content

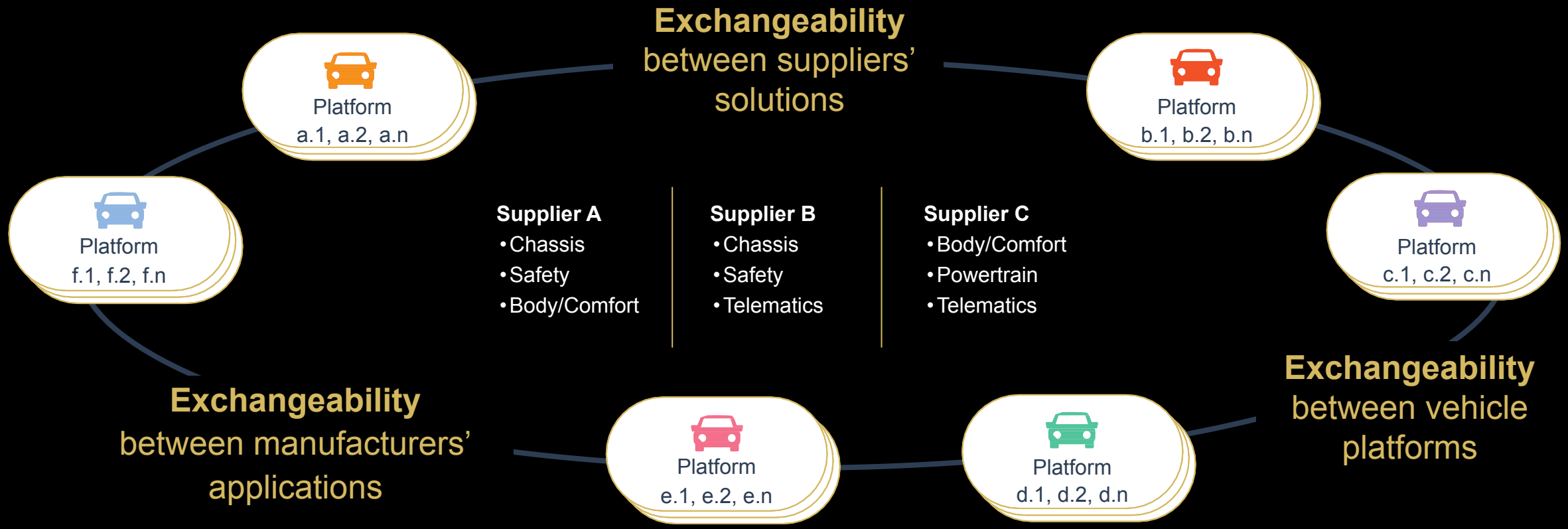
- Automotive Software Industry
- **Autosar Overview**
- CAN Network
- Classic Autosar
  - Autosar Architecture
  - Autosar Concepts and Methodology
  - Autosar Application and RTE
  - Autosar Communication stack
  - Autosar IO stack
  - Autosar Memory Stack and RTE
  - Autosar OS and RTE
  - Autosar Watchdog Stack
  - Cybersecurity and Autosar Crypto Stack

AUTOSAR Overview

- AUTOSAR Vision
- AUTOSAR Goal
- AUTOSAR Partner
- AUTOSAR Platforms

AUTOSAR Vision

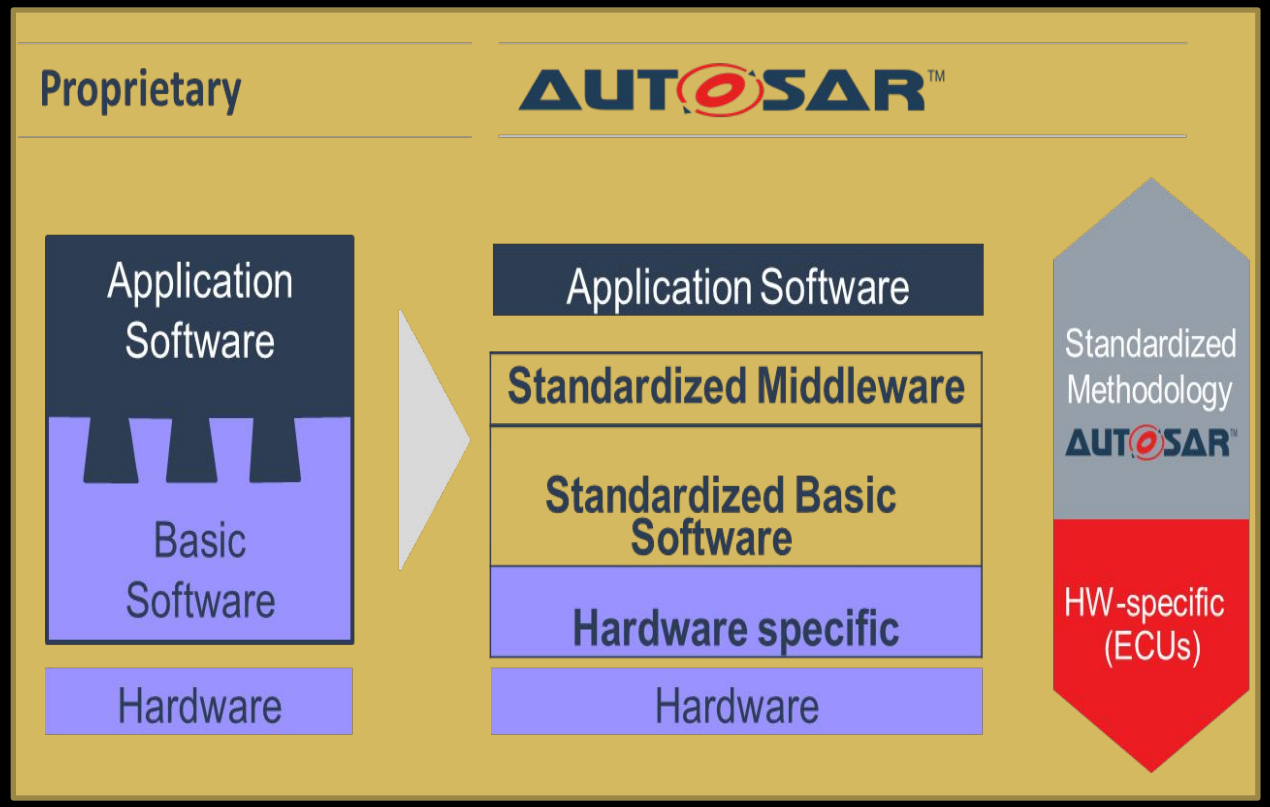
AUTOSAR aims to improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between OEMs and suppliers.



AUTOSAR Goal

AUTOSAR aims to standardize the software architecture ECUs.

- Hardware and software – widely independent of each other.
- Development can be decoupled (through abstraction) by horizontal layers, reducing development time and costs.
- Reuse of software enhances quality and efficiency



AUTOSAR Partners

More Than 280 Partners

9 Core Partners

BMW Group · BOSCH · Continental · DAIMLER · Ford · GM · PSA GROUPE · TOYOTA · VOLKSWAGEN AKTIENGESELLSCHAFT

56 Premium Partners

altran · APTIV · APTJ · arm · Baidu 百度 · BlackBerry · Capgemini · C22 · DASSAULT SYSTEMES · Deloitte. · dSPACE · eSOL · ETAS · Fraunhofer · Great Wall · Green Hills SOFTWARE · HCL · HELLA · HONDA · HUAWEI · HYUNDAI MOTOR GROUP · Infineon · intel · intron 英恒 · itk · JTEKT · KPIT · L&T Technology Services · LEAR CORPORATION · KLUXOFT · MathWorks · Mentor Graphics · MITSUBISHI ELECTRIC · NEC · NISSAN · NXP · Panasonic · RENAULT · RENESAS · SCSK · sodius · STMicroelectronics · SUMITOMO ELECTRIC · TATA · TATA ELXSI · TATA CONSULTANCY SERVICES · thyssenkrupp · Valeo · VECTOR · veoneer · VOLVO · WIND · ZF

2 Strategic Partners

LG Life's Good

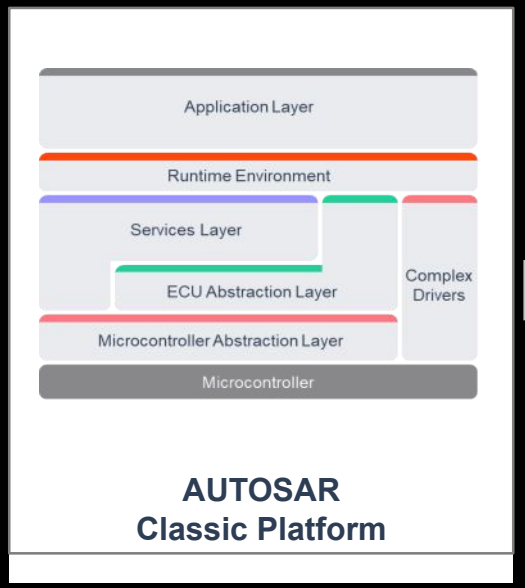
DENSO Crafting the Core

51 Development Partners

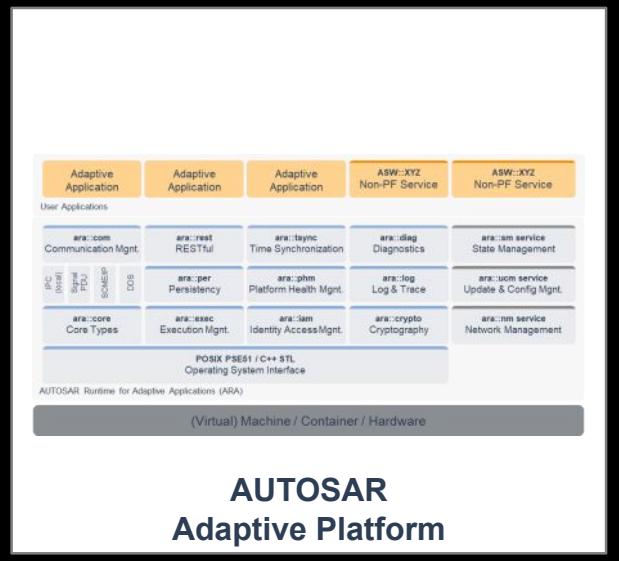
ABUP 艾拉比 · avelabs · BASELABS · basicworx · C2A · clarinox · codeplay · eodycore · eJad · LABS · EVOLVE · evo · excelfore · FPT Software · Freetech · GAIO TECHNOLOGY · HANECS · HINGE · INTREPID CONTROL SYSTEMS · INCHRON · IPCamp · iSYSTEM · ixia · KRONO-SAFE · LAUTERBACH · MATRICKZ · NCES · REACH AUTOMOTIVE TECHNOLOGY · nordsys · OPENSYNERGY · pls · PopcornSAR · PRQA · RTAE · rti · RTst · SAFERIDE · SEMIDRIVE · seneos · softing · SYSGO · TTTech · VALIDAS · VAULT MICRO · VEROLT · WAFER SPACE

+ 144 Associate Partners
+ 24 Attendees

AUTOSAR And Automotive Challenges



AUTOSAR
Classic Platform

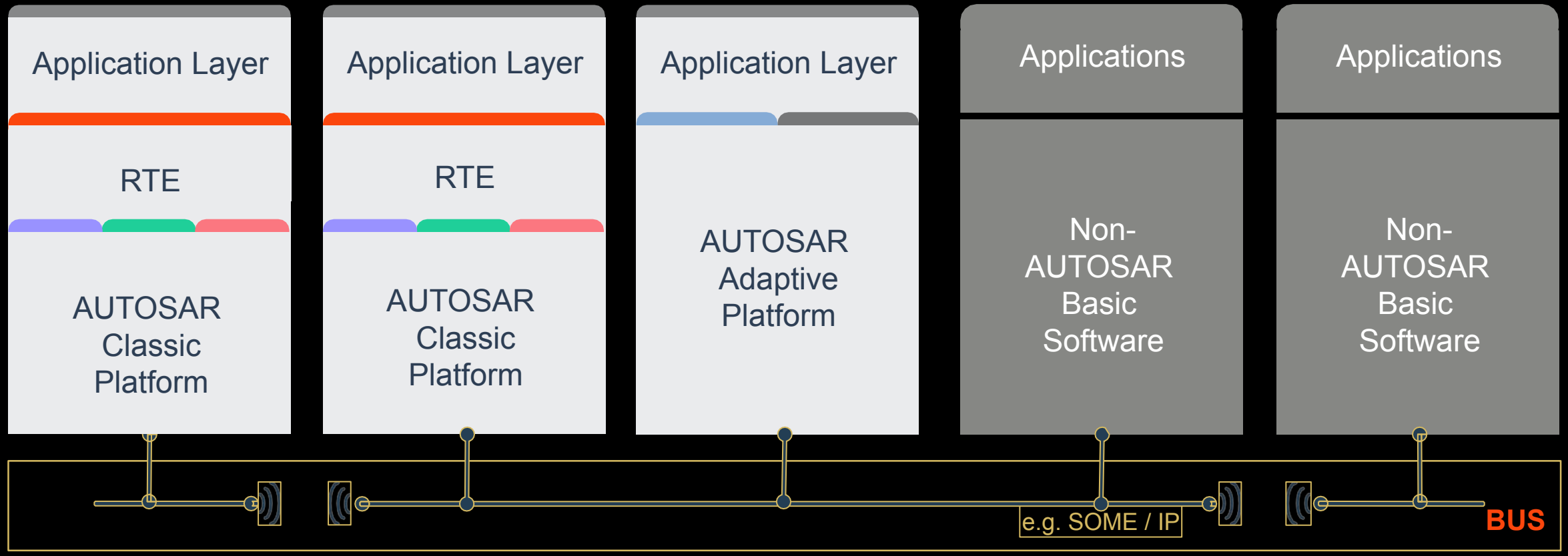


AUTOSAR
Adaptive Platform

Infotainment

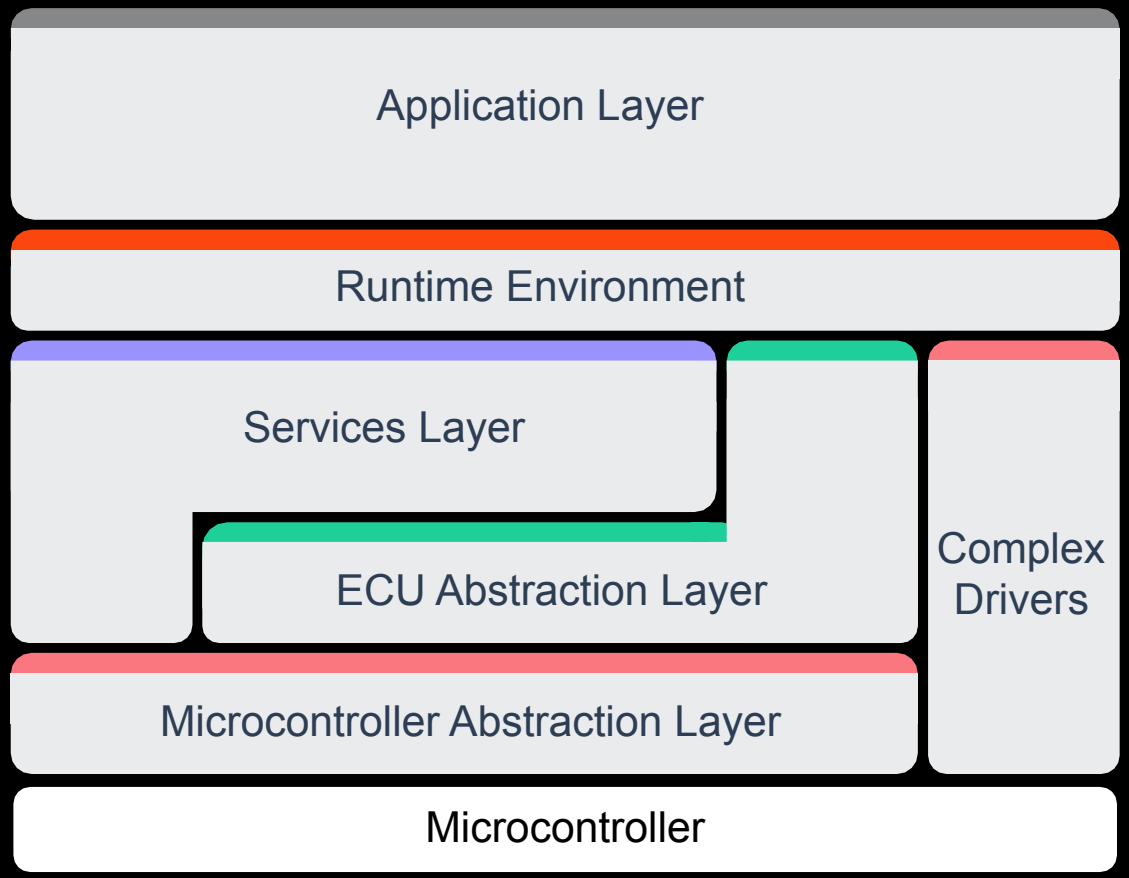
Real time Requirements	High, in the range of micro-sec	Mid, in the range of milli-sec	Low, in the range of sec
Safety Criticality	High, up to ASIL-D	High, at least ASIL-B	Low, QM
Computing power	Low, ~ 1000 DMIPs	High, >20.000 DMIPs	High, ~ 10.000 DMIPs

AUTOSAR in a Vehicle Network



Common Bus Interface Specification

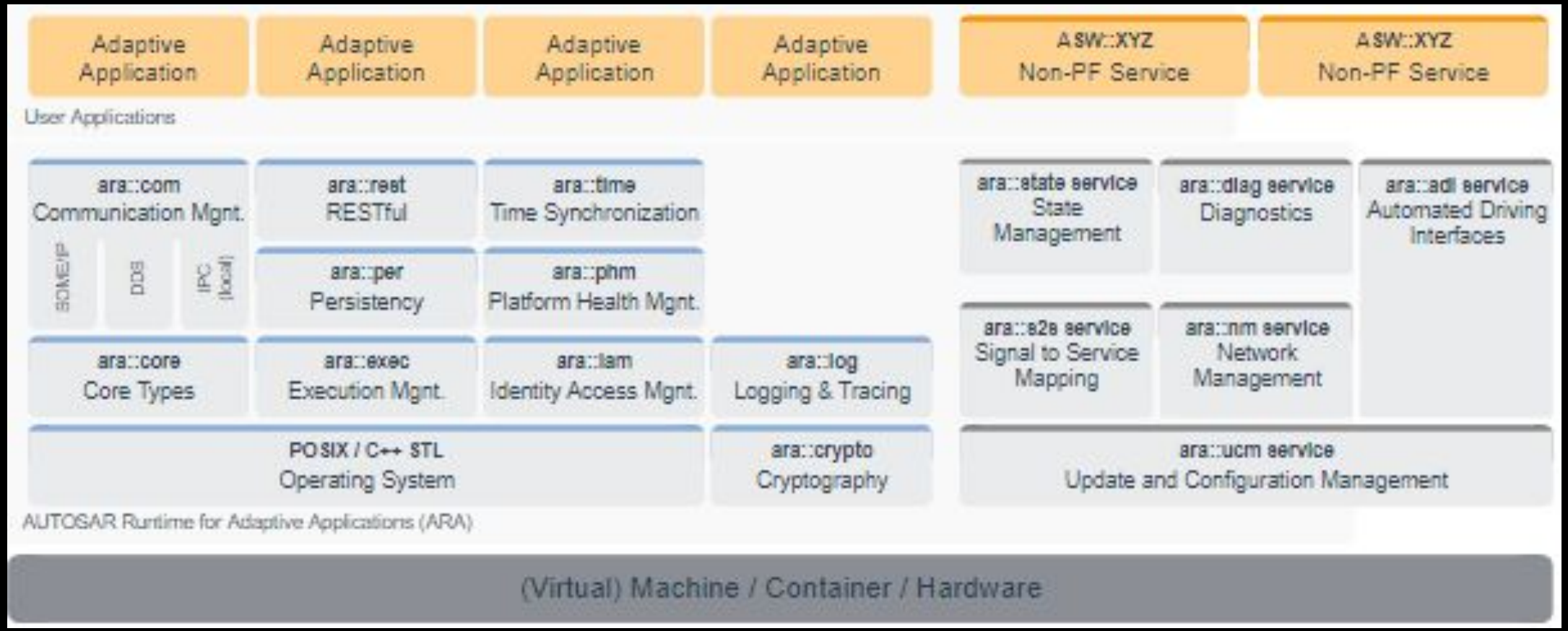
AUTOSAR Classic Platform



The layered architecture of the classic platform basically supports

- Hardware abstraction
- Scheduling OS tasks
- Communication between applications on the same hardware and over the network
- Diagnosis and diagnostic services
- Safety- and Security Services

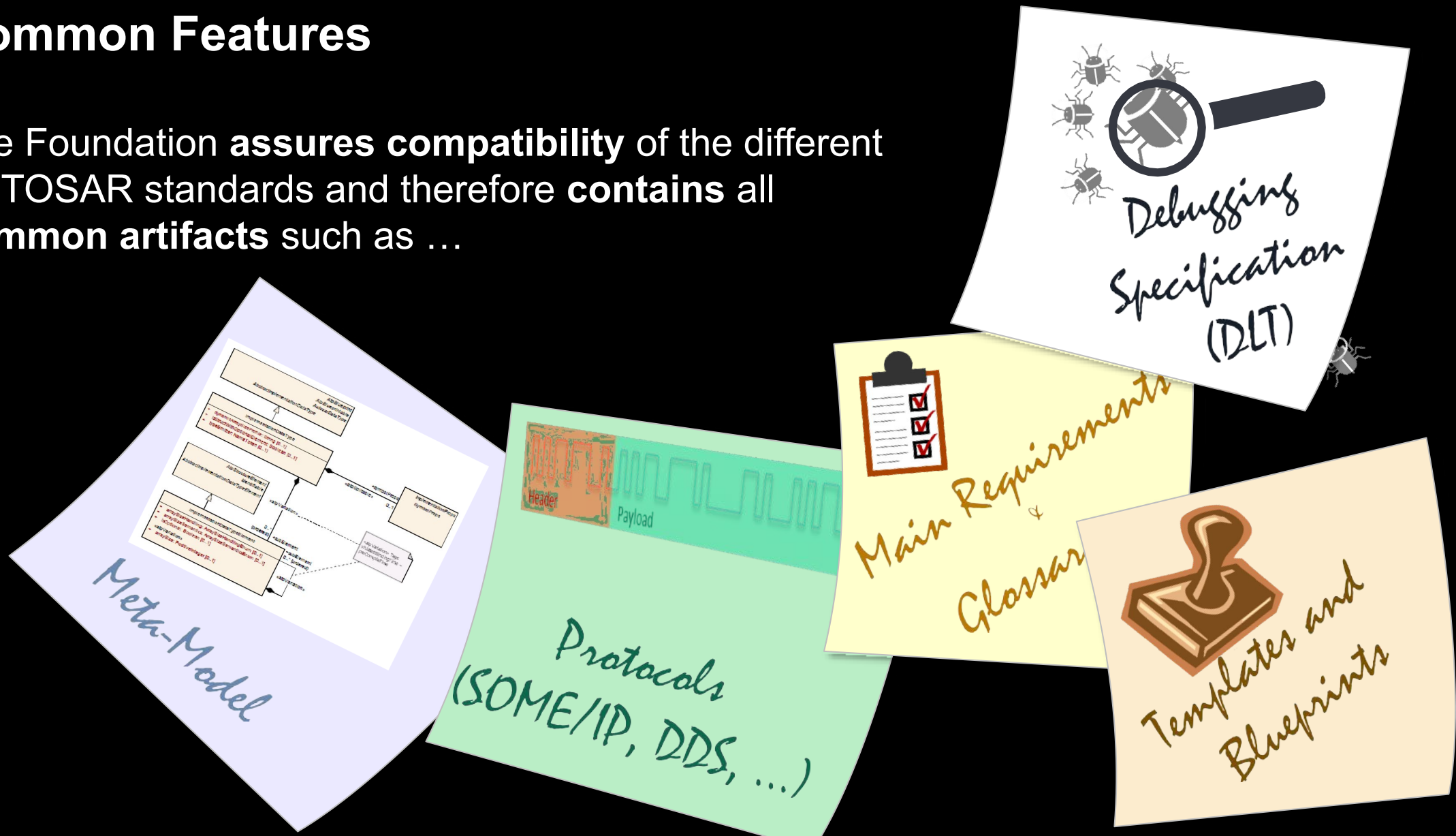
AUTOSAR Adaptive Platform



AUTOSAR Foundation

Common Features

The Foundation **assures compatibility** of the different AUTOSAR standards and therefore **contains** all **common artifacts** such as …



Course Content

- Automotive Software Industry
- Autosar Overview
- CAN Network
- **Classic Autosar**
  - Autosar Architecture
  - Autosar Concepts and Methodology
  - Autosar Application and RTE
  - Autosar Communication stack
  - Autosar IO stack
  - Autosar Memory Stack and RTE
  - Autosar OS and RTE
  - Autosar Watchdog Stack
  - Cybersecurity and Autosar Crypto Stack

- AUTOSAR Vision
- AUTOSAR Goal
- AUTOSAR Partner
- AUTOSAR Platforms

Classic Platform

AUTOSAR Application Scope

•AUTOSAR is dedicated for Automotive ECUs.

•Such ECUs have the following properties:

- Strong interaction with hardware (sensors and actuators)
- Connection to vehicle networks like CAN, LIN, FlexRay or Ethernet
- Microcontrollers (typically 16 or 32 bit) with limited resources of computing power and memory (compared with enterprise solutions)
- Real Time System
- Program execution from internal or external flash memory.

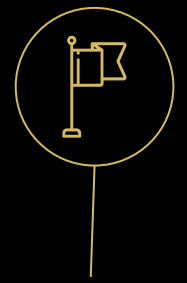
AUTOSAR Extensibility

•The AUTOSAR Software Architecture is a generic approach:

- Standard modules can be extended in functionality, while still being compliant,
- Their configuration has to be considered in the automatic Basic SW configuration process!
- Non-standard modules can be integrated into AUTOSAR-based systems as Complex Drivers and
- Further layers cannot be added.

Classic Autosar

Architecture



Methodology

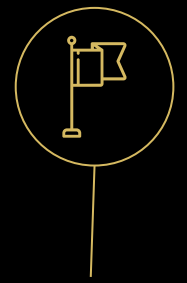
Concept

Course Content

- Automotive Software Industry
- Autosar Overview
- CAN Network
- Classic Autosar
  - **Autosar Architecture**
  - Autosar Concepts and Methodology
  - Autosar Application and RTE
  - Autosar Communication stack
  - Autosar IO stack
  - Autosar Memory Stack and RTE
  - Autosar OS and RTE
  - Autosar Watchdog Stack
  - Cybersecurity and Autosar Crypto Stack

Classic Autosar

Architecture



Methodology

Concept

AUTOSAR Architecture

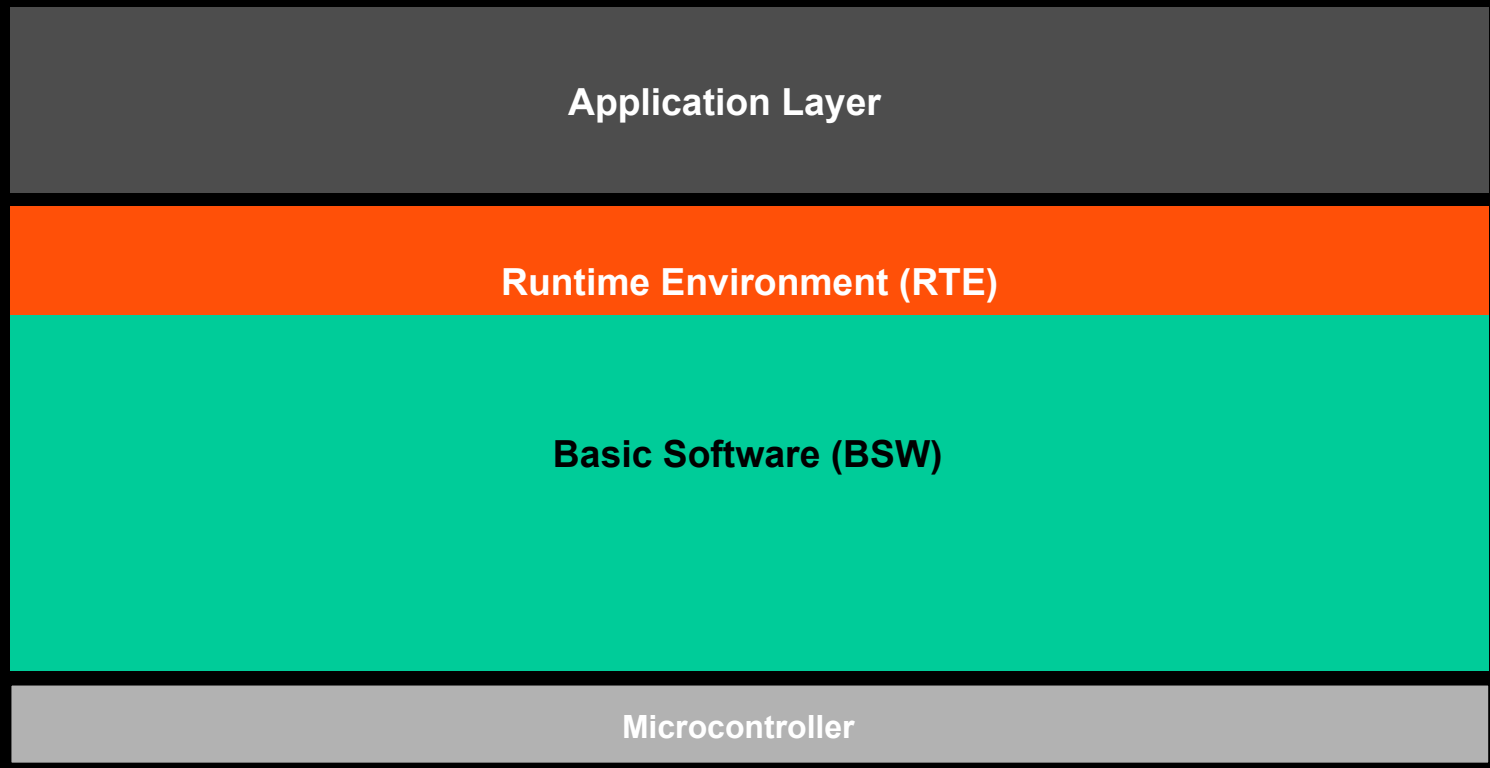
- BSW
- RTE, SWCs & Lib.
- AUTOSAR Stacks

Classic Platform

AUTOSAR Architecture

- Distinguishes on the highest abstraction level between three software layers:
  - Application (SWCs)
  - Runtime Environment (RTE)
  - Basic Software (BSW)

- The main concept of the standardized ECU software architecture is the separation of hardware-independent application software and hardware-oriented basic software by means of the software abstraction layer RTE.



AUTOSAR Architecture

- On the upper side of the RTE

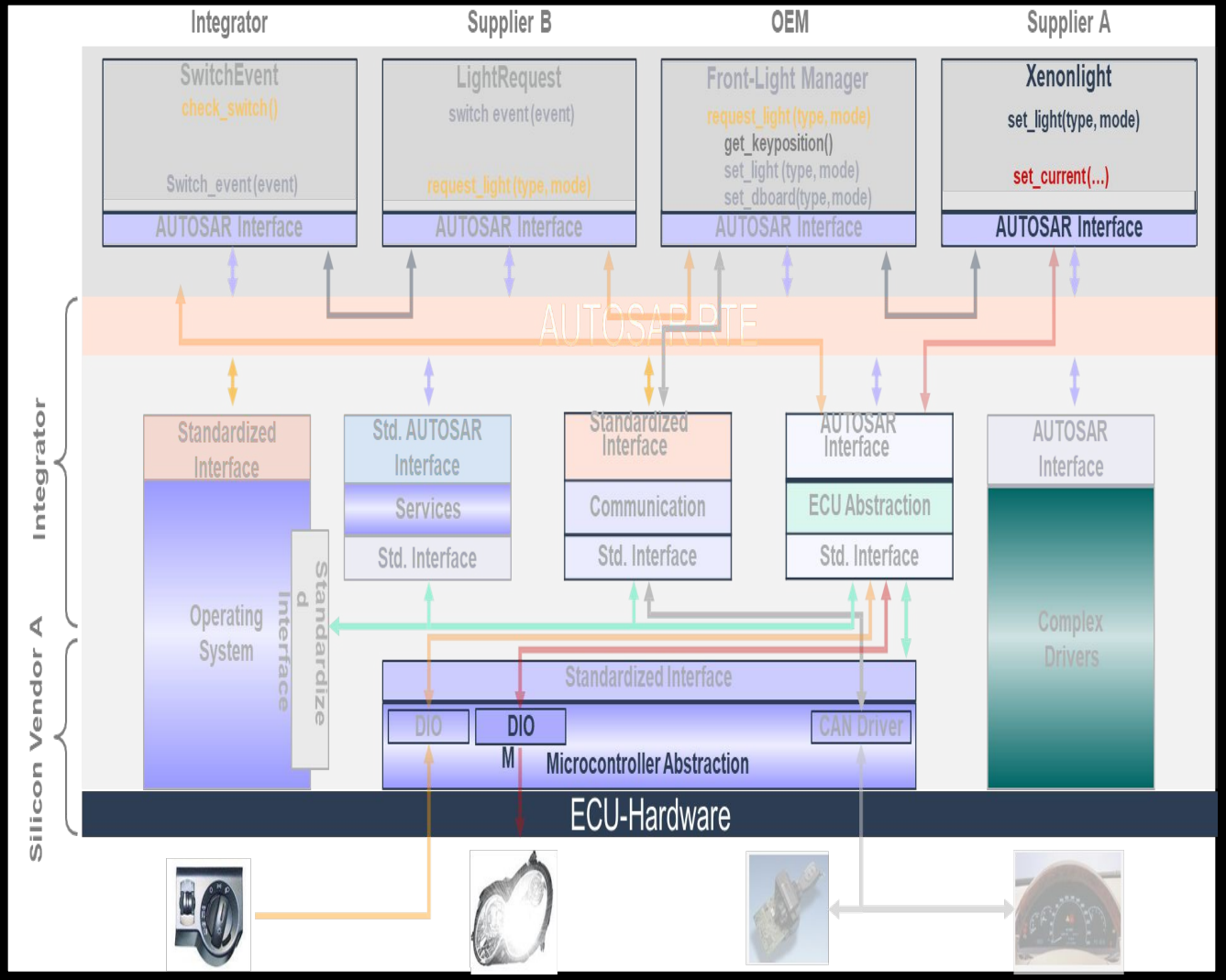
  Abstraction layer enables the development of OEM-specific and competitive software applications (SWCs).

- On the lower side of the RTE

  Enables the standardization and OEM-independence of basic software.

AUTOSAR Architecture

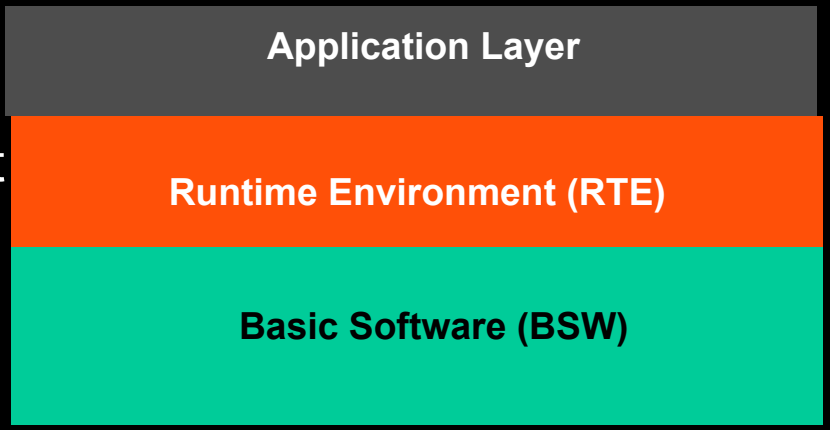
- AUTOSAR increase the scalability of ECU software for several car lines and variants.
- Distribute applications (functional software modules) across ECUs, and the ability to integrate software modules from different sources.



AUTOSAR Architecture

Runtime Environment (RTE)

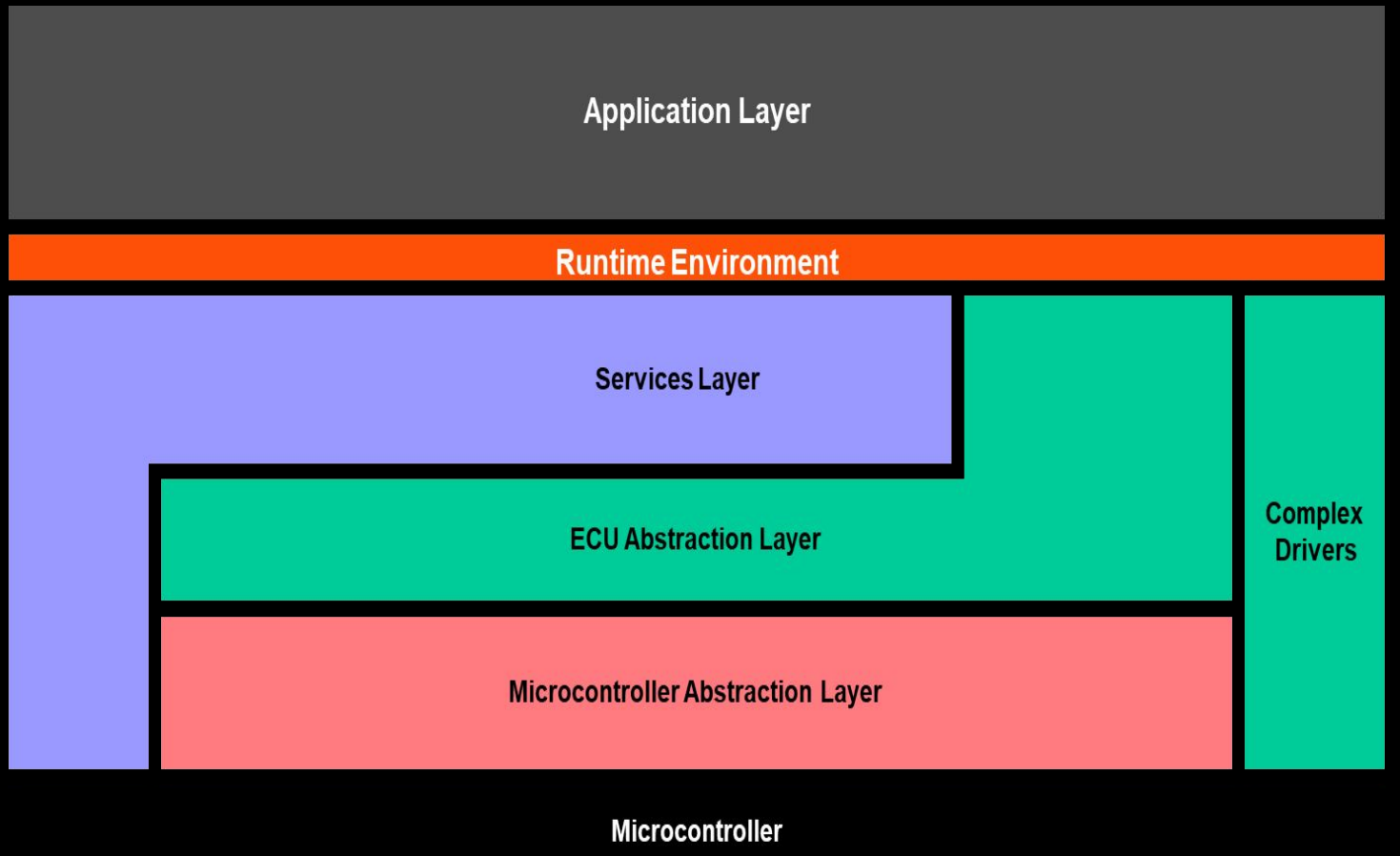
- The separation of the application layer from the basic software
- Includes the control of the data exchange between these layers.
- Forms the basis for a component-oriented, hardware-independent software structure on application level, with software components (SWCs) as individual units.
- It is possible to develop SWCs without specific knowledge of the hardware used or planned
- Flexibly relocate existing SWCs to ECUs during development.



AUTOSAR Architecture

Basic Software Layers

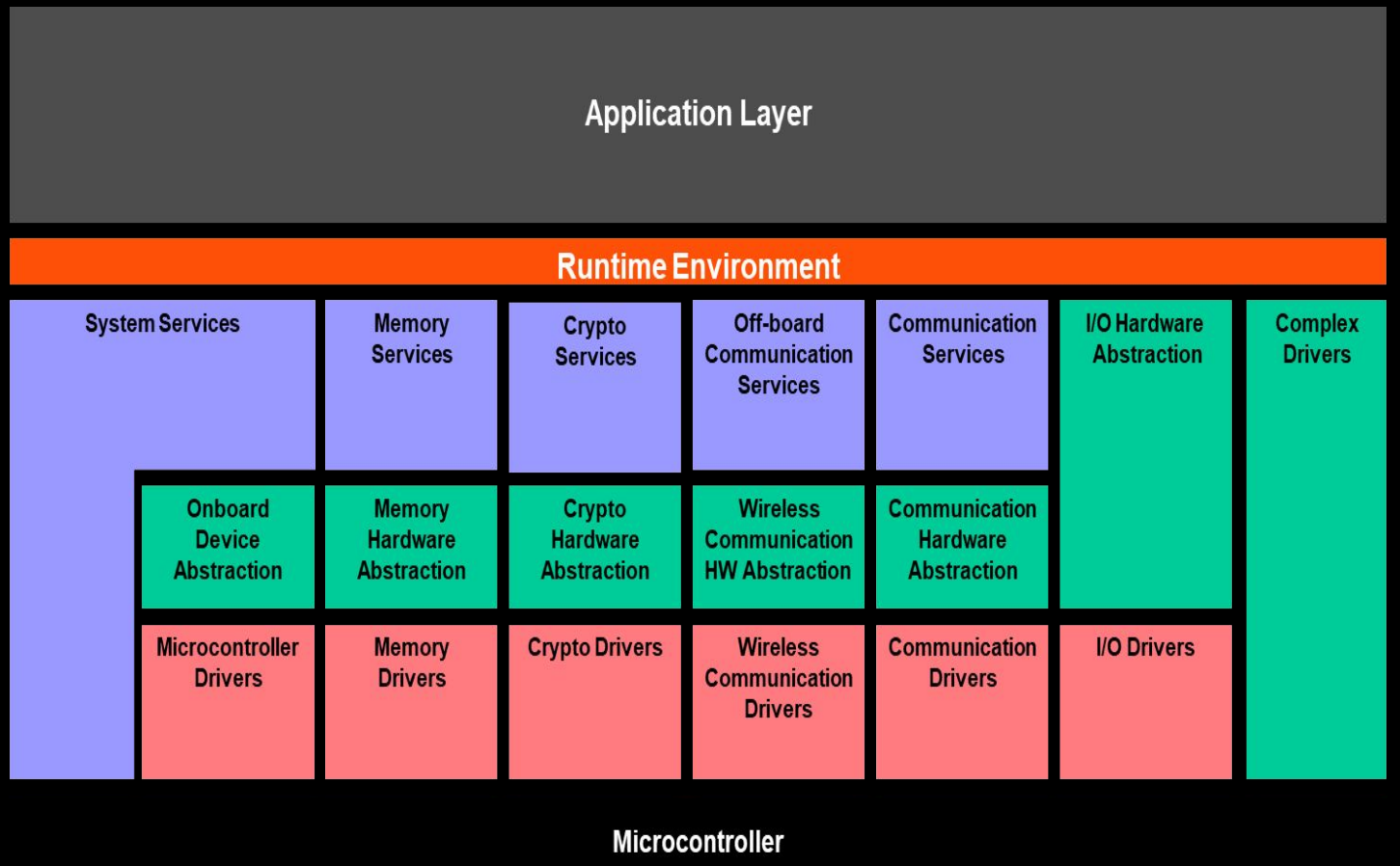
- Divided into the following layers:
  - Services
  - ECU abstraction
  - Microcontroller abstraction



AUTOSAR Architecture

Basic Software Stacks

- The Basic Software Layers are further divided into functional groups (6 Stacks)
  - System
  - Memory
  - Communication
  - Input Output
  - Crypto
  - Off-board communication



01

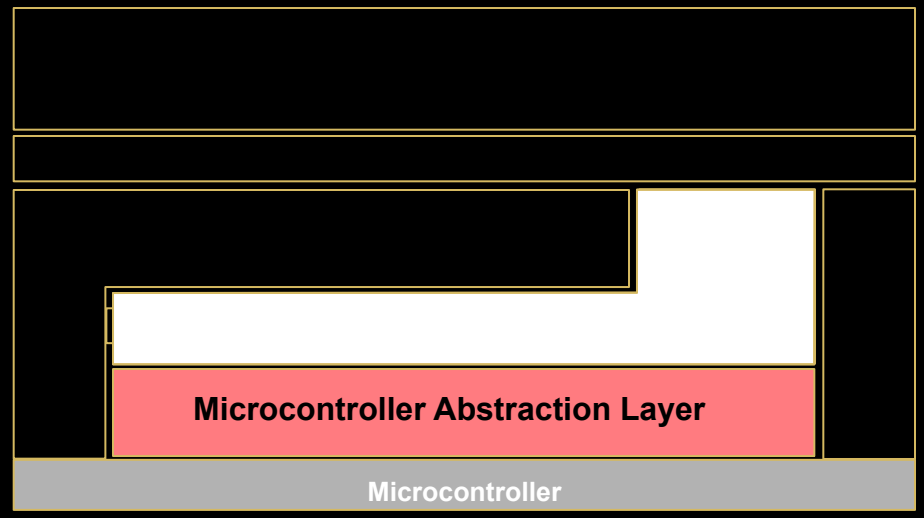
AUTOSAR BSW

Basic Software Layers

Microcontroller Abstraction Layer (MCAL)

- The is the lowest software layer of the Basic Software.
- It contains internal drivers, which are software modules with direct access to the µC and internal peripherals.
- Task

  Make higher software layers independent of µC
- Properties
  - Implementation: µC dependent
  - Upper Interface: standardized and µC independent



Basic Software Layers

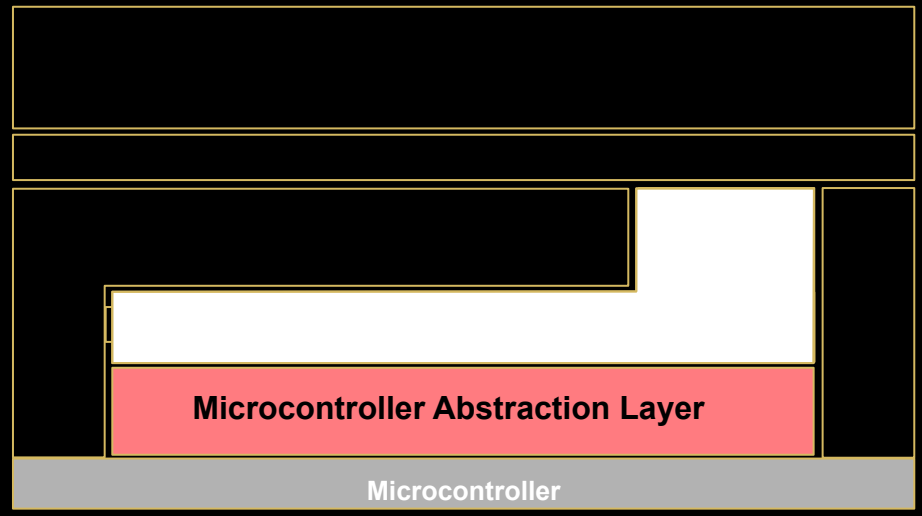
Microcontroller Abstraction Layer (MCAL)

Internal Driver

- Contains the functionality to control and access an internal peripheral.

  Internal EEPROM, Internal CAN controller and Internal ADC

- Located in the MCAL

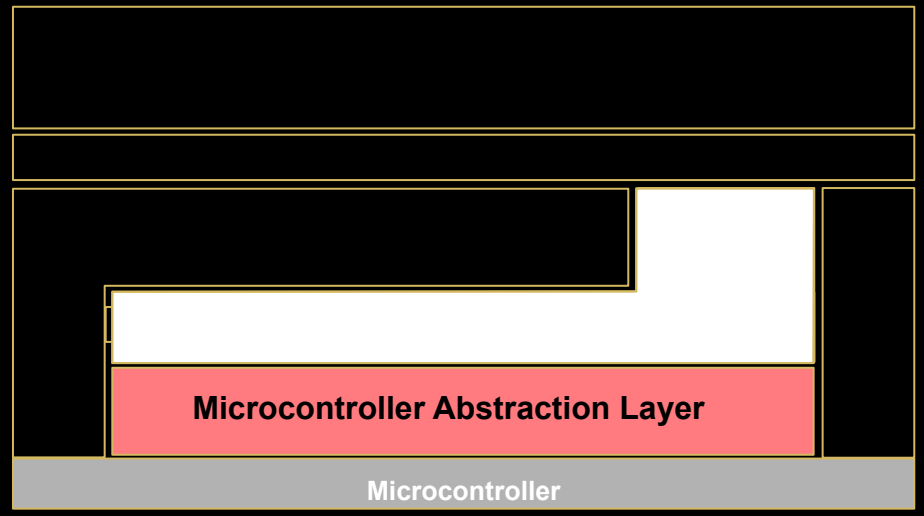


Basic Software Layers

Microcontroller Abstraction Layer (MCAL)

Handler

- A is a specific interface which controls the concurrent, multiple and asynchronous access of one or multiple clients to one or more drivers. I.e. it performs buffering, queuing, arbitration, multiplexing.
- The handler does not change the content of the data.
- Handler functionality is often incorporated in the driver or interface (e.g. SPIHandlerDriver).

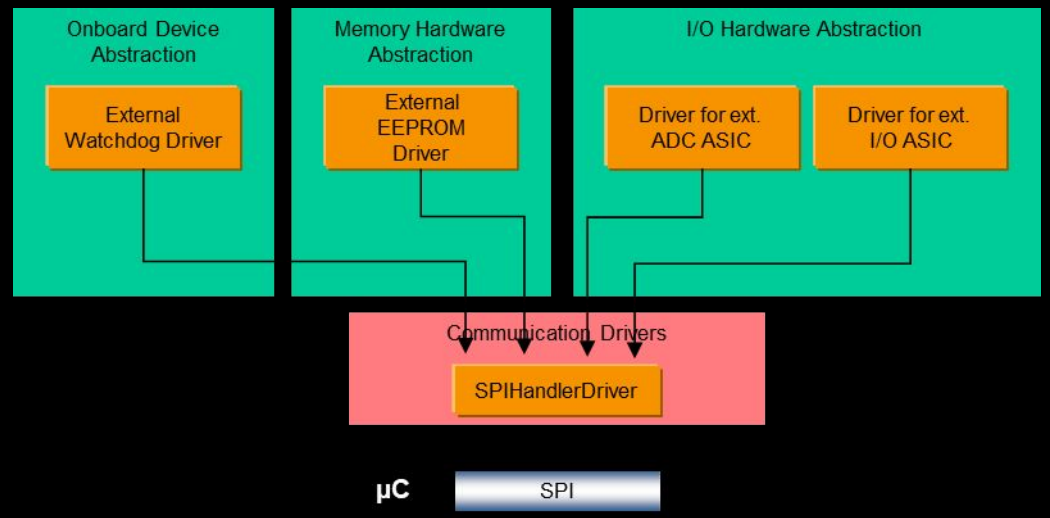


Basic Software Layers

Microcontroller Abstraction Layer (MCAL)

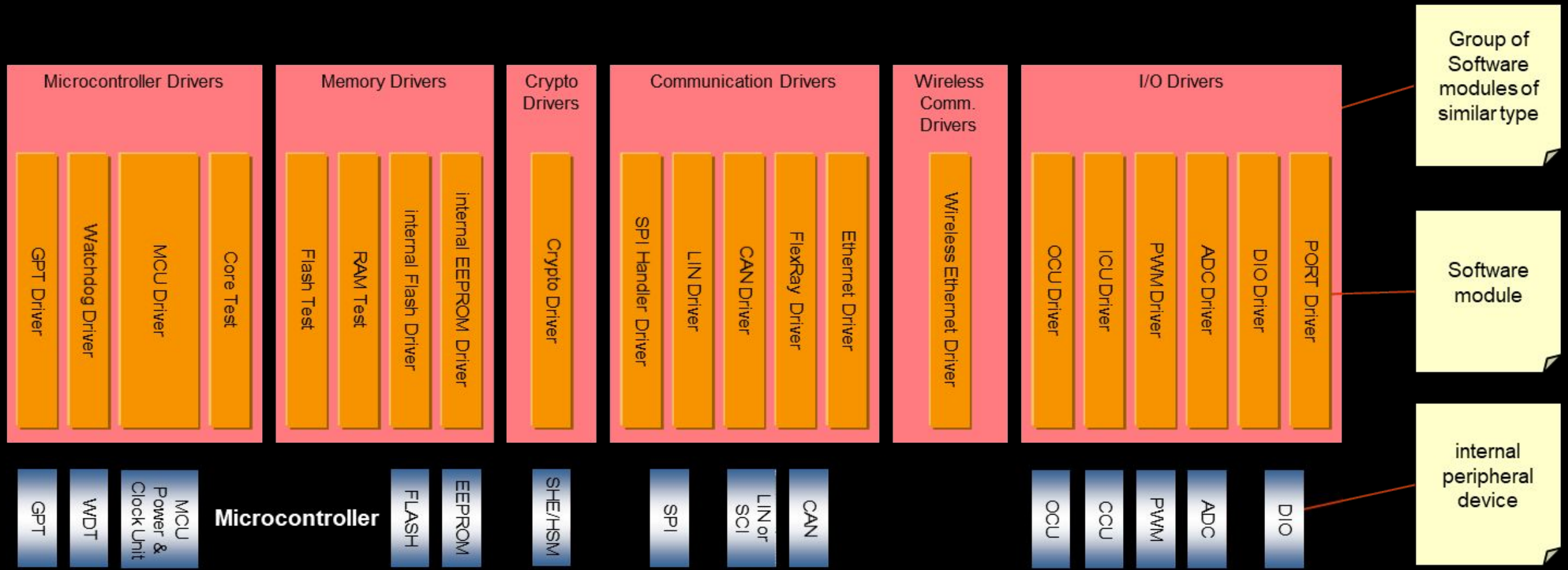
Handler

- The SPIHandlerDriver allows concurrent access of several clients to one or more SPI busses.
- To abstract all features of a SPI microcontroller pins dedicated to Chip Select, those shall directly be handled by the SPIHandlerDriver.
- That means those pins shall not be available in DIO Driver.



Basic Software Layers

Microcontroller Abstraction Layer (MCAL)

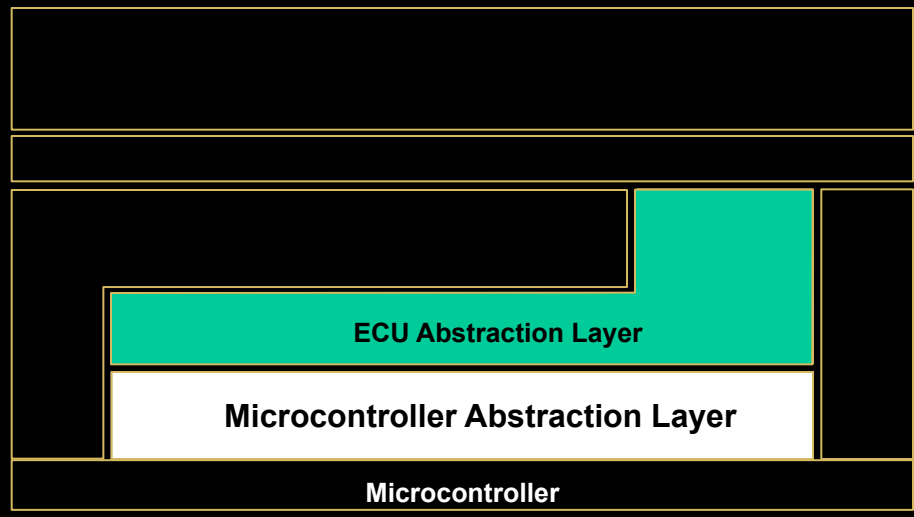


Basic Software Layers

ECU Abstraction Layer (ECAL)

- Contains drivers for external devices.
- Interfaces the MCAL drivers by offering APIs for access to peripherals and devices regardless of their location (µC internal/external) and their connection to the µC (port pins, type of interface)
- Task

  Make higher software layers independent of ECU hardware layout
- Properties
  - Implementation: µC independent, ECU hardware dependent
  - Upper Interface: µC and ECU hardware independent



Basic Software Layers

ECU Abstraction Layer (ECAL)

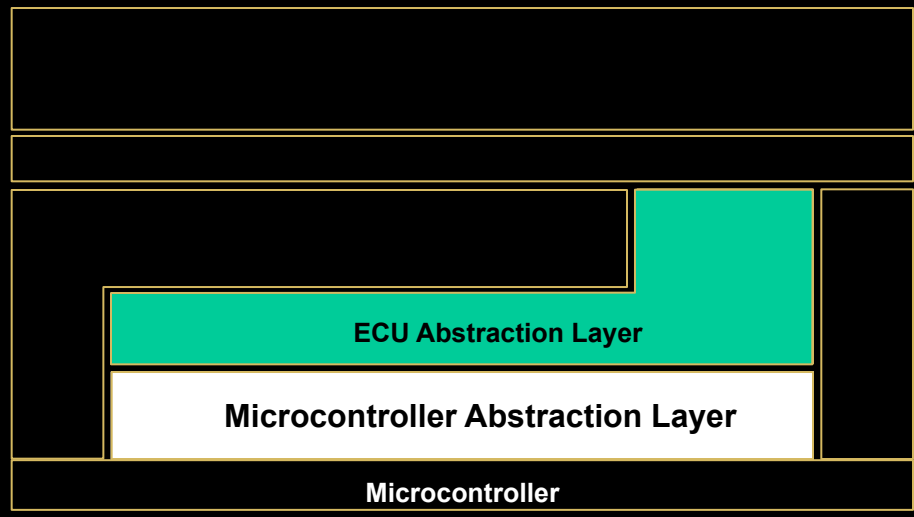
External Driver

- Contains the functionality to control and access an external device.

  External EEPROM, External watchdog, External flash

- Accesses the external device via drivers MCAL.

  A driver for an external EEPROM with SPI interface accesses the external EEPROM via the handler/driver for the SPI bus.



- Exception:

The drivers for memory mapped external devices (e.g. external flash memory) may access the microcontroller directly.

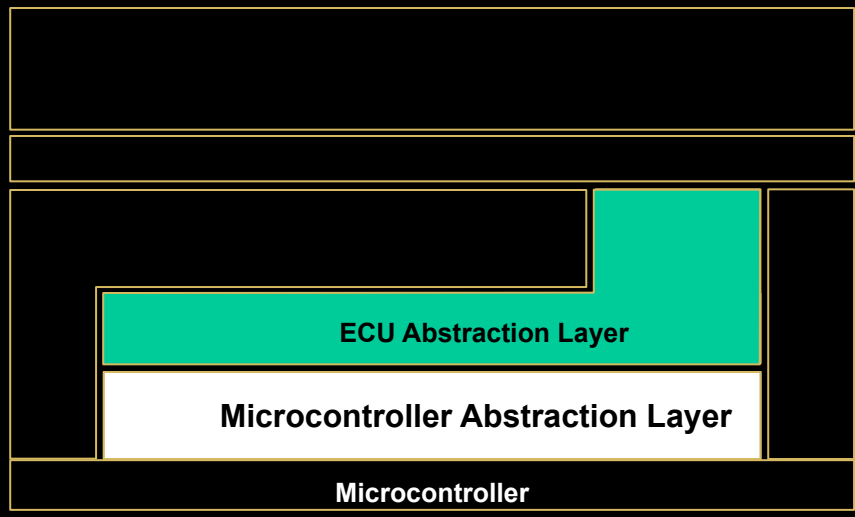
Those external drivers are located in the Microcontroller Abstraction Layer because they are microcontroller dependent.

Basic Software Layers

ECU Abstraction Layer (ECAL)

Interface

- Contains the functionality to abstract from modules which are architecturally placed below them.
- E.g., an interface module which abstracts from the hardware realization of a specific device.
- It provides a generic API to access a specific type of device independent on the number of existing devices of that type and independent on the hardware realization of the different devices.
- The interface does not change the content of the data.

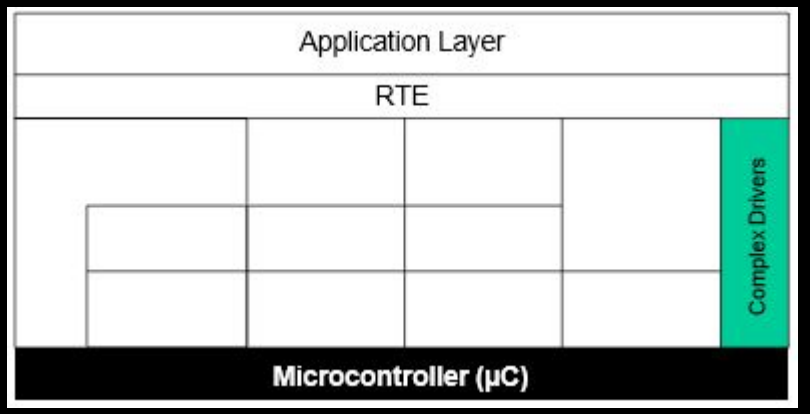


- Example: an interface for a CAN communication system provides a generic API to access CAN communication networks independent on the number of CAN Controllers within an ECU and independent of the hardware realization (on chip, off chip).

Basic Software Layers

Complex Drivers

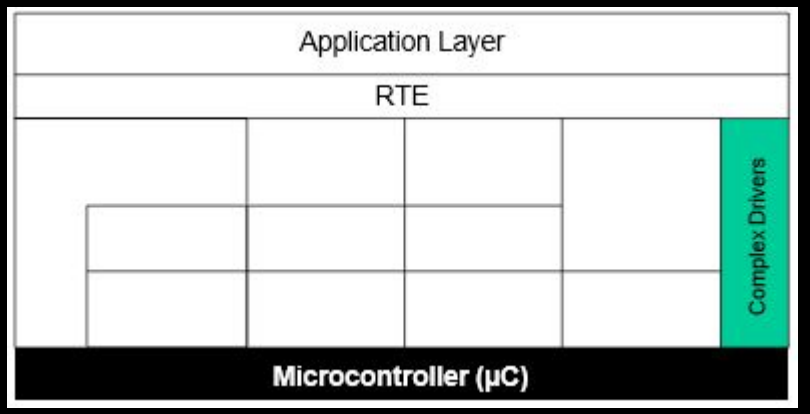
- Spans from the hardware to the RTE
- implements non-standardized functionality within the basic software stack
- Fulfill the special functional and timing requirements for handling complex sensors and actuators
- Task
  - Provide the possibility to integrate special purpose functionality, e.g. drivers for devices:
    - which are not specified within AUTOSAR,
    - with very high timing constrains or
    - for migration purposes etc.



Basic Software Layers

Complex Drivers

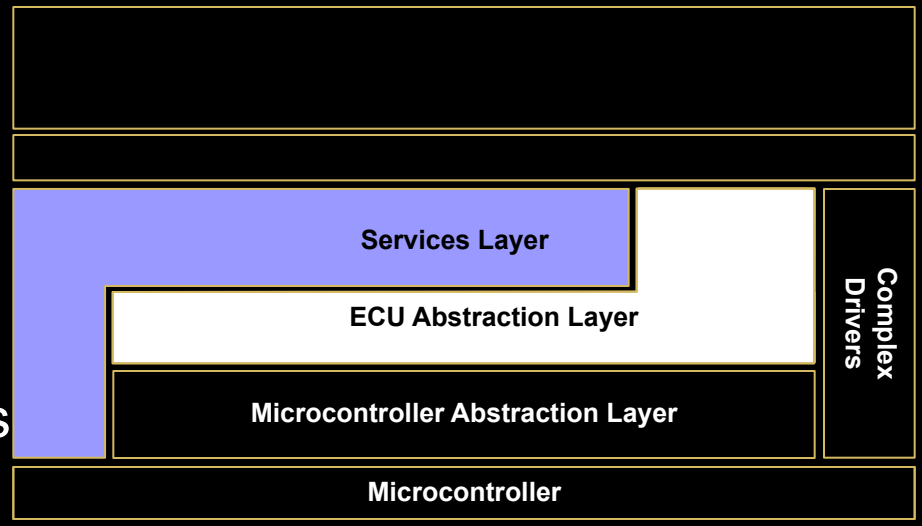
- Properties
  - Implementation: might be application, µC and ECU hardware dependent
  - Upper Interface: might be application, µC and ECU hardware dependent
  - Upper Interface to SW-Cs: specified and implemented according to AUTOSAR (AUTOSAR interface)
  - Lower interface: restricted access to Standardized Interfaces



Basic Software Layers

Services Layer

- The highest layer of the Basic Software which also applies for its relevance for the application software:
- While access to I/O signals is covered by the ECU Abstraction Layer, the Services Layer offers:
  - Operating system functionality
  - Vehicle network communication and management services
  - Memory services (NVRAM management)
  - Diagnostic Services (including UDS communication, error memory and fault treatment)
  - ECU state management, mode management
  - Logical and temporal program flow monitoring (Wdg manager)



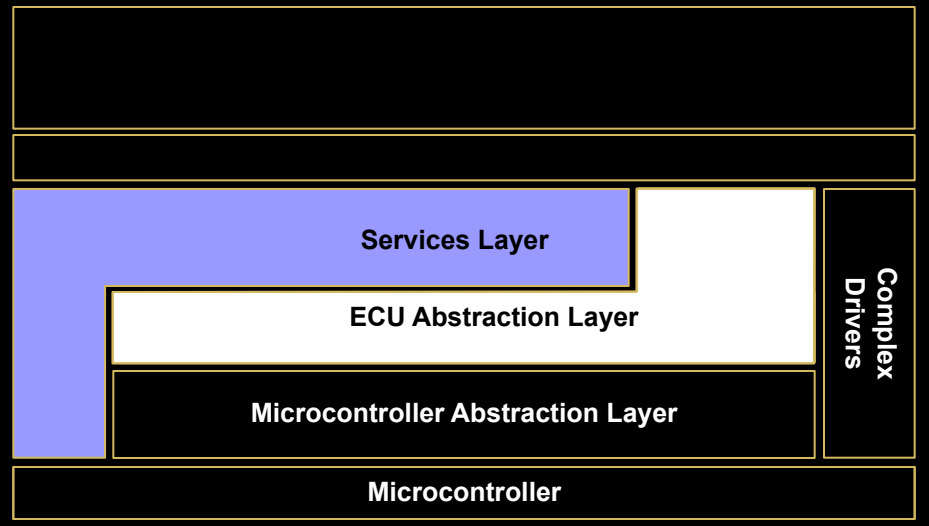
Basic Software Layers

Services Layer

- Task

  Provide basic services for applications, RTE and basic software modules.

- Properties
  - Implementation: mostly µC and ECU hardware independent
  - Upper Interface: µC and ECU hardware independent



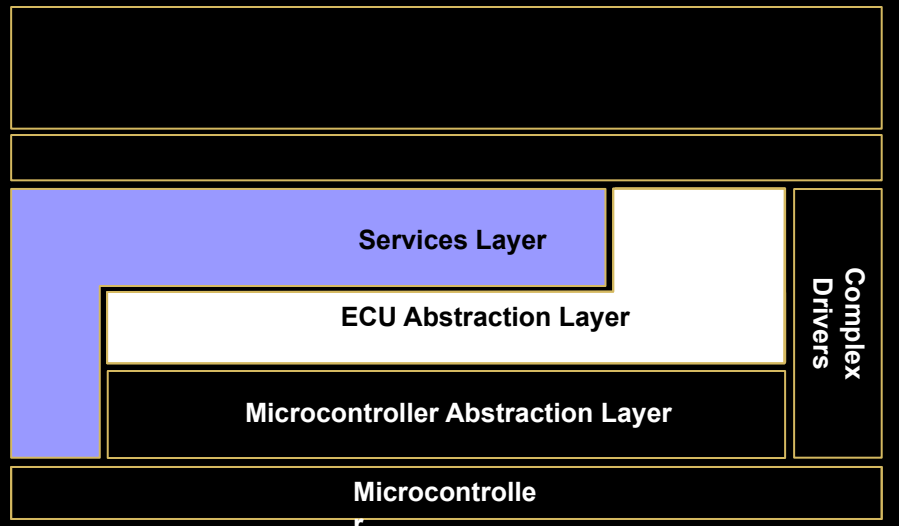
Basic Software Layers

Services Layer

Manager

- Offers specific services for multiple clients.
- It is needed in all cases where pure handler functionality is not enough to abstract from multiple clients.
- Besides handler functionality, a manager can evaluate and change or adapt the content of the data.
- Example: The NVRAM manager manages the concurrent access to internal and/or external memory devices like flash and EEPROM memory.

  It also performs distributed and reliable data storage, data checking, provision of default values etc.



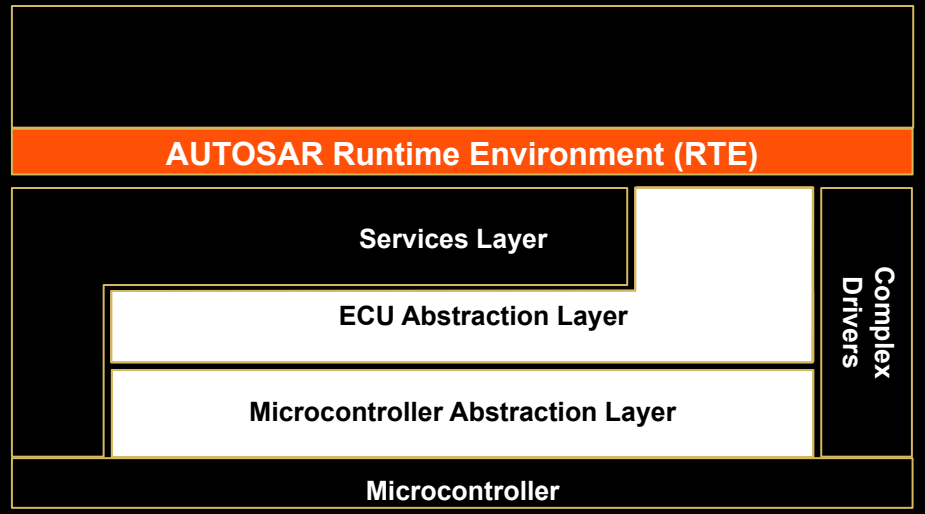
02

AUTOSAR RTE, SWCs & Libraries

Runtime Environment (RTE)

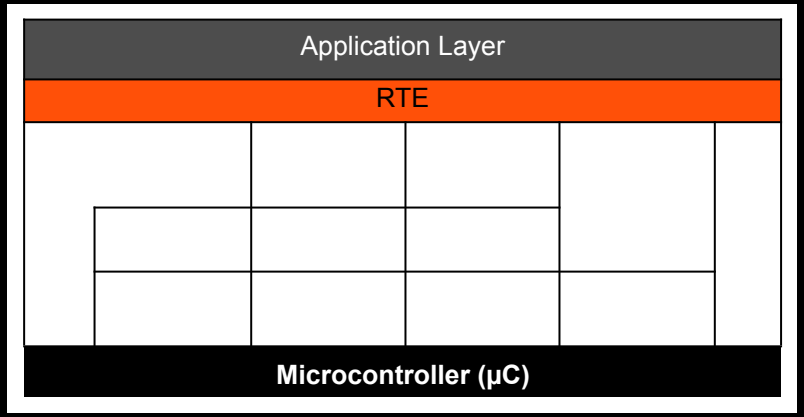
- Providing communication services to SWCs.
- Above the RTE the software architecture style changes from “layered“ to “component style“.
- The SWCs communicate with other components (inter and/or intra ECU) and/or services via the RTE.
- Task

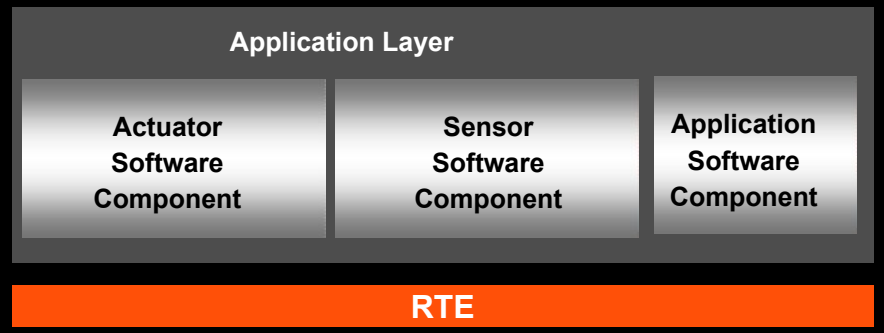
  Make AUTOSAR Software Components independent from the mapping to a specific ECU.
- Properties
  - Implementation: ECU and application specific (generated individually for each ECU)
  - Upper Interface: completely ECU independent



Autosar Software Component

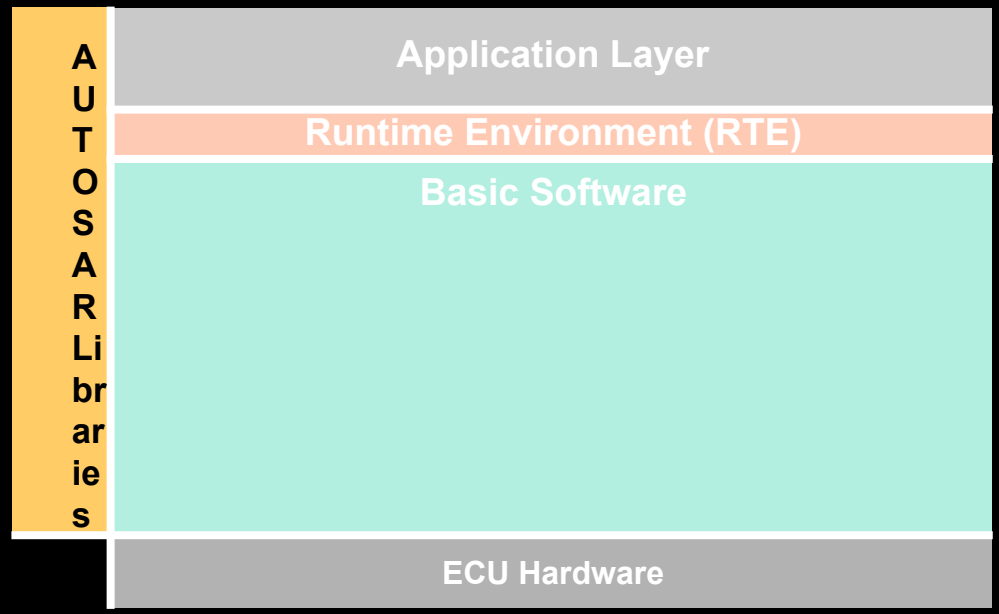
- The Sensor/Actuator SWC is for sensor evaluation and actuator control.
- Though not belonging to the AUTOSAR Basic Software, it is described here due to its strong relationship to local signals.
- It has been decided to locate the Sensor/Actuator SW Components above the RTE for integration reasons (standardized interface implementation and interface description).
- Because of their strong interaction with raw local signals, relocatability is restricted.





Libraries

- Libraries are a collection of functions for related purposes
- Can be called by BSW modules (that including the RTE), SW-Cs, libraries or integration code.
- Run in the context of the caller in the same protection environment
- Can only call libraries
- Are re-entrant
- Do not have internal states
- Do not require any initialization
- are synchronous, i.e. they do not have wait points



The following libraries are specified within AUTOSAR:

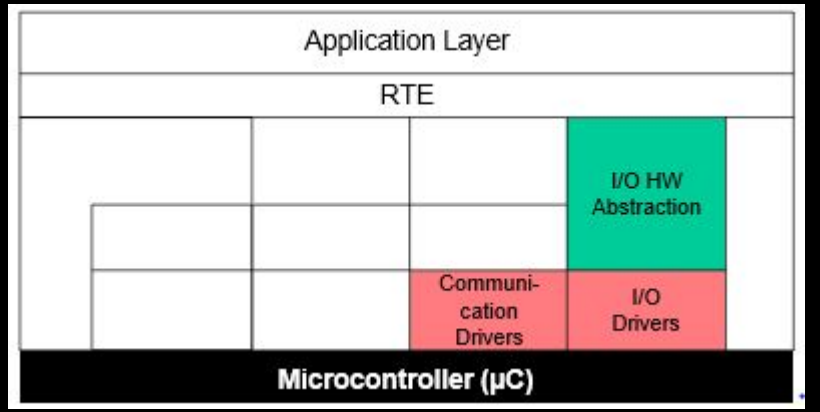
- Fixed point mathematical,
- Floating point mathematical,
- Interpolation for fixed point data,
- Interpolation for floating point data,
- Bit handling,
- E2E communication,
- CRC calculation,
- Extended functions (e.g. 64bits calculation, filtering, etc.)

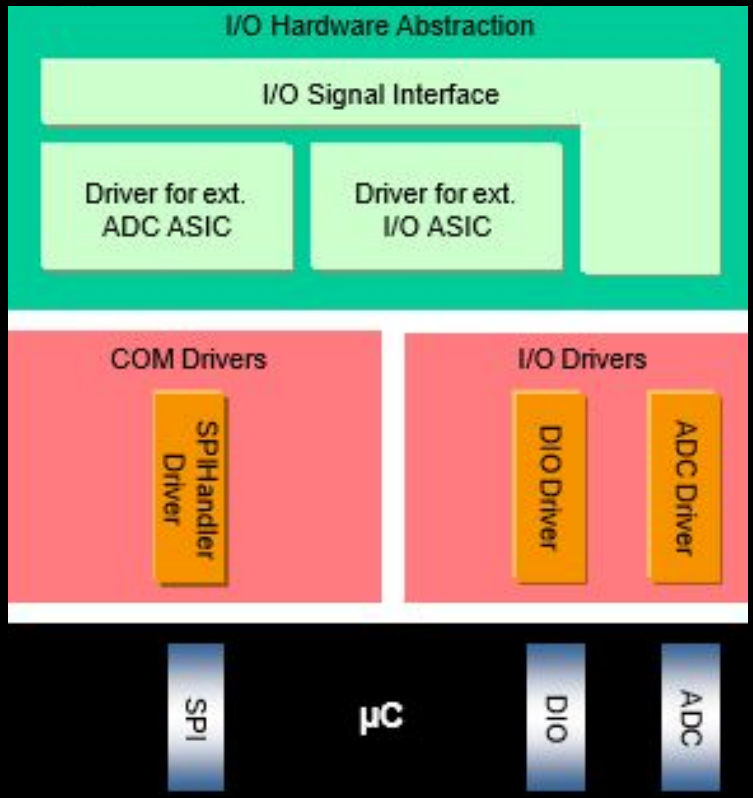
03

AUTOSAR Stacks

IO Stack

- Represent I/O signals as they are connected to the ECU hardware (e.g. current, voltage, frequency).
- Hide ECU hardware and layout properties from higher software layers.



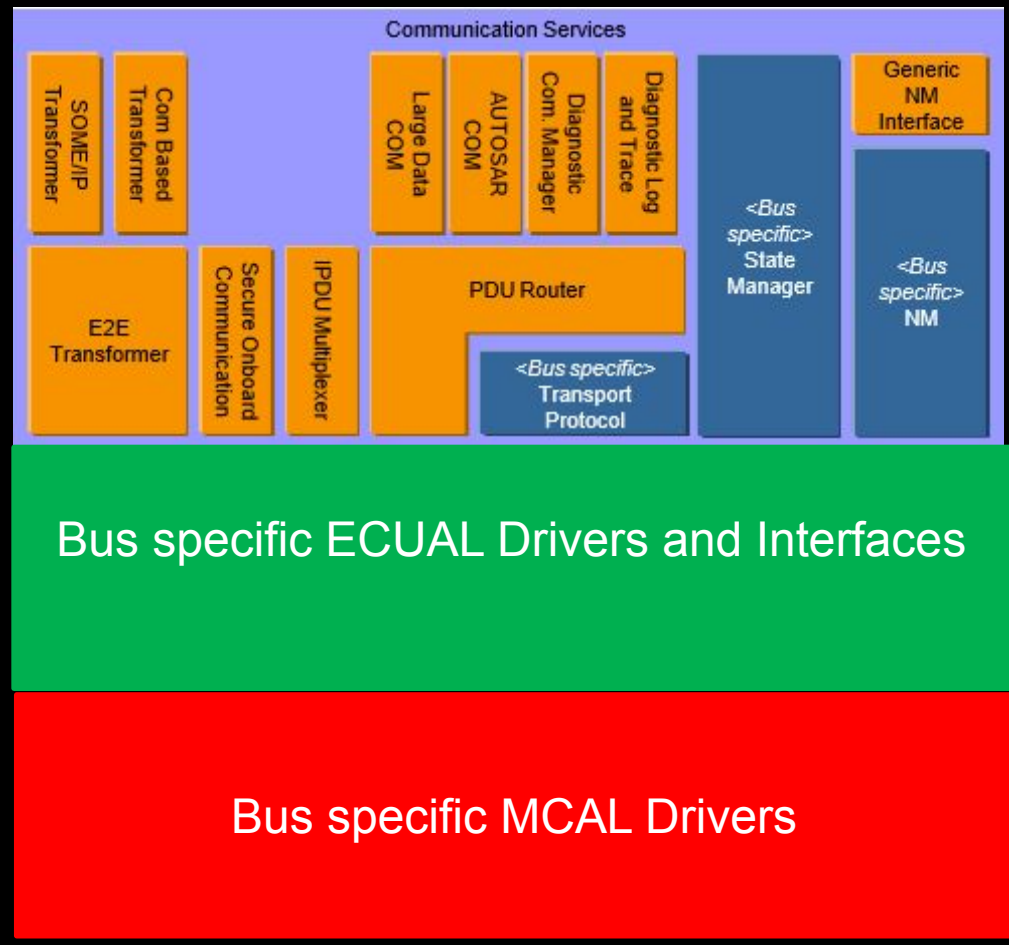


Communication Stack

- Divide into
  - Data Path
  - Network management
- Provide
  - Uniform interface to the vehicle network for communication.
  - Uniform services for network management Provide uniform interface to the vehicle network for
  - Diagnostic communication

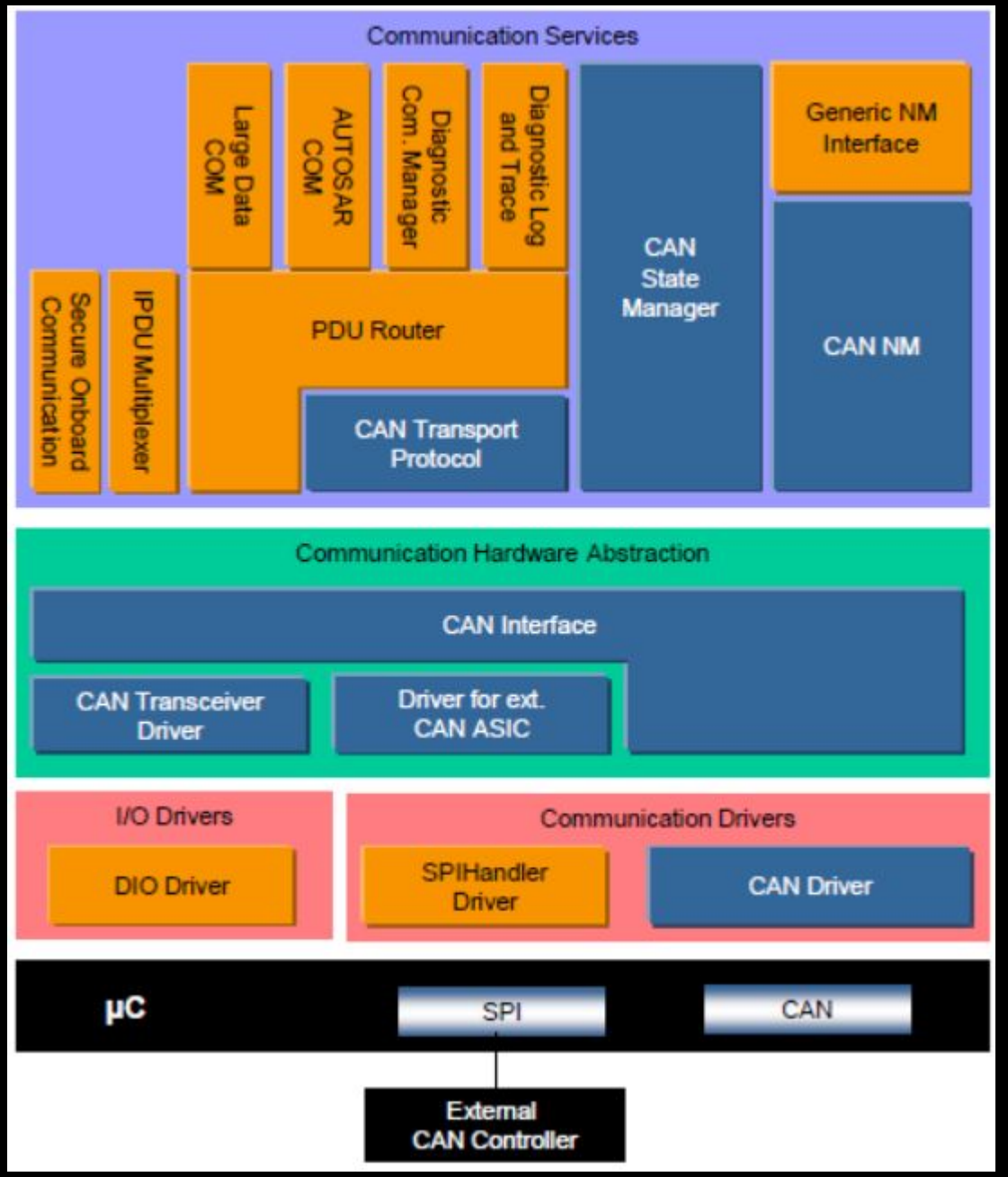
    Diagnostic Communication Manager(DCM)

    Provides a common API for diagnostic services
  - Hide protocol and message properties from the application.



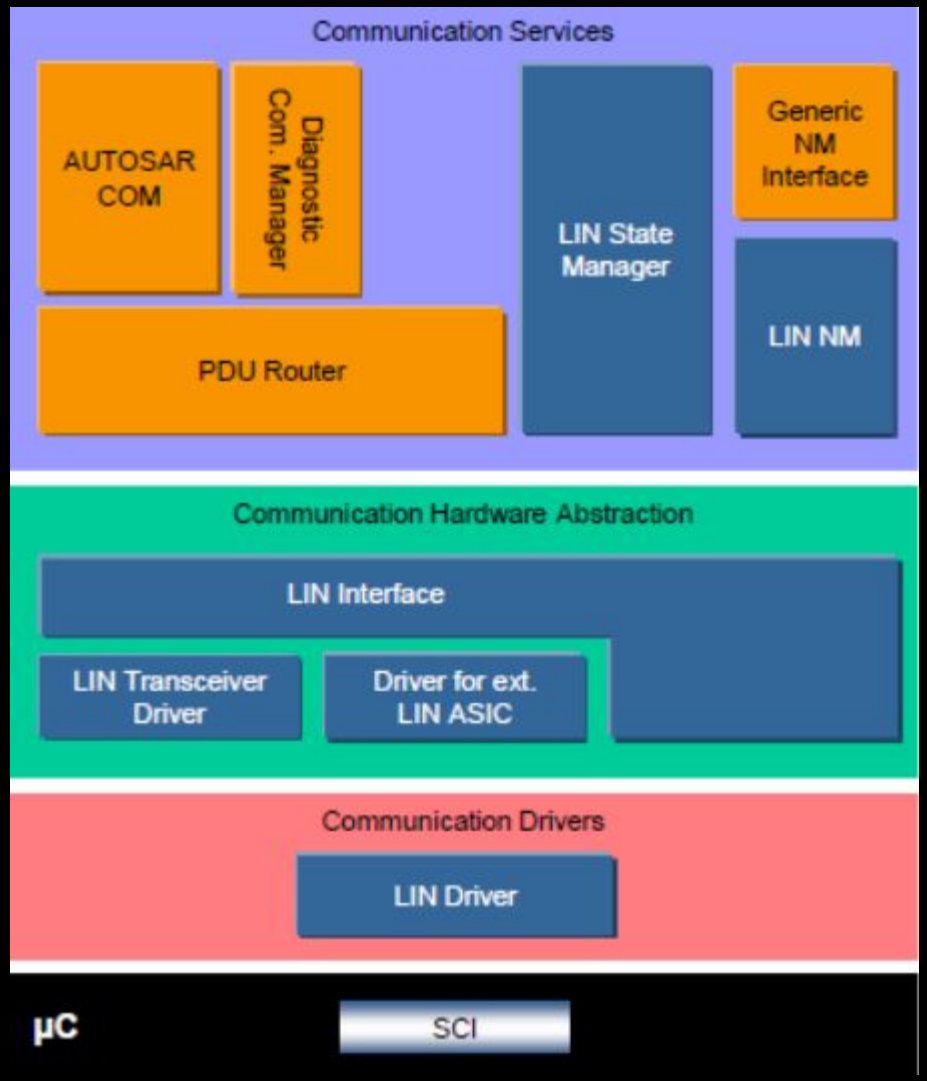
Communication Stack

CAN Stack



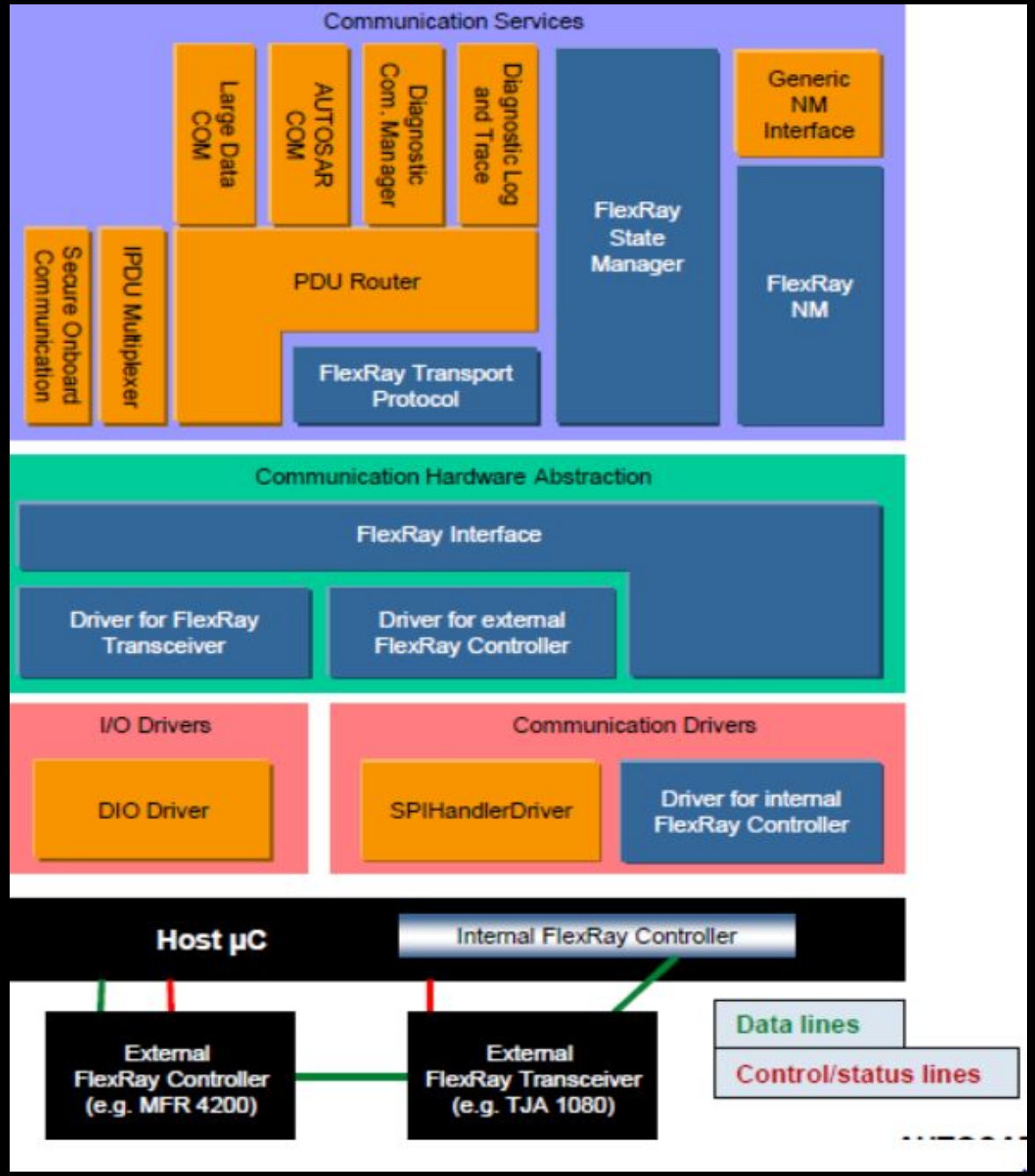
Communication Stack

LIN Stack



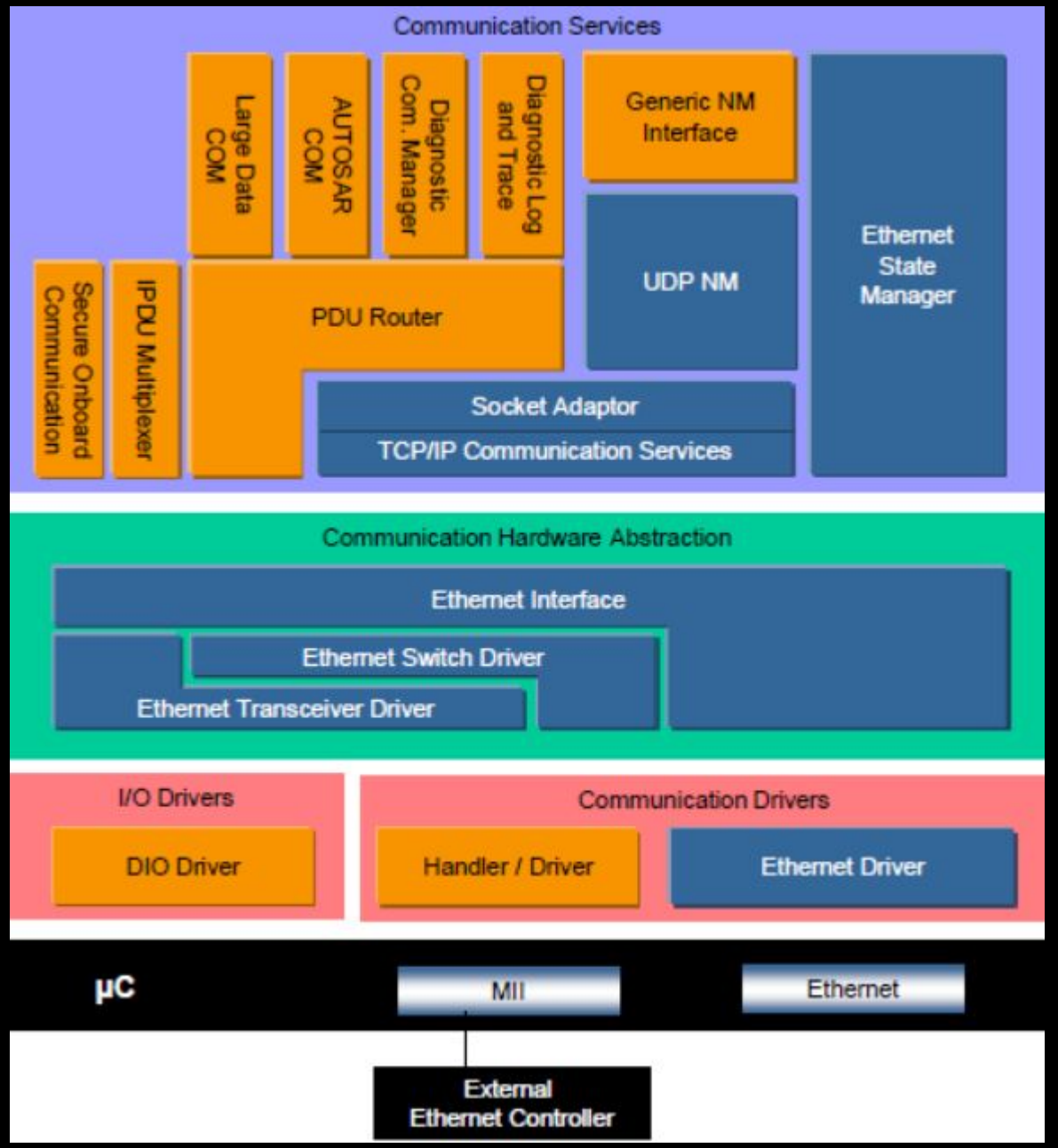
Communication Stack

FlexRay Stack

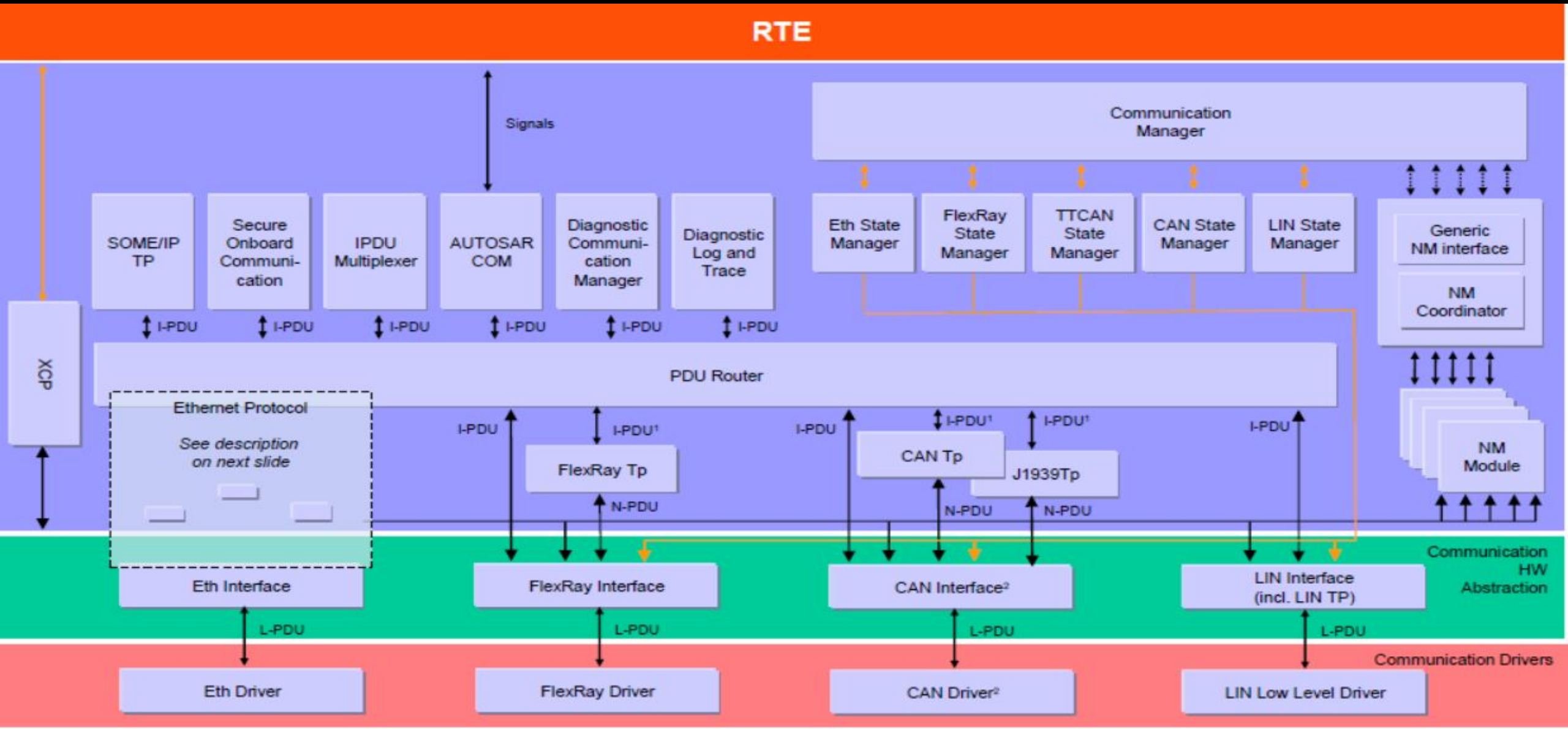


Communication Stack

Ethernet Stack



Communication Stack

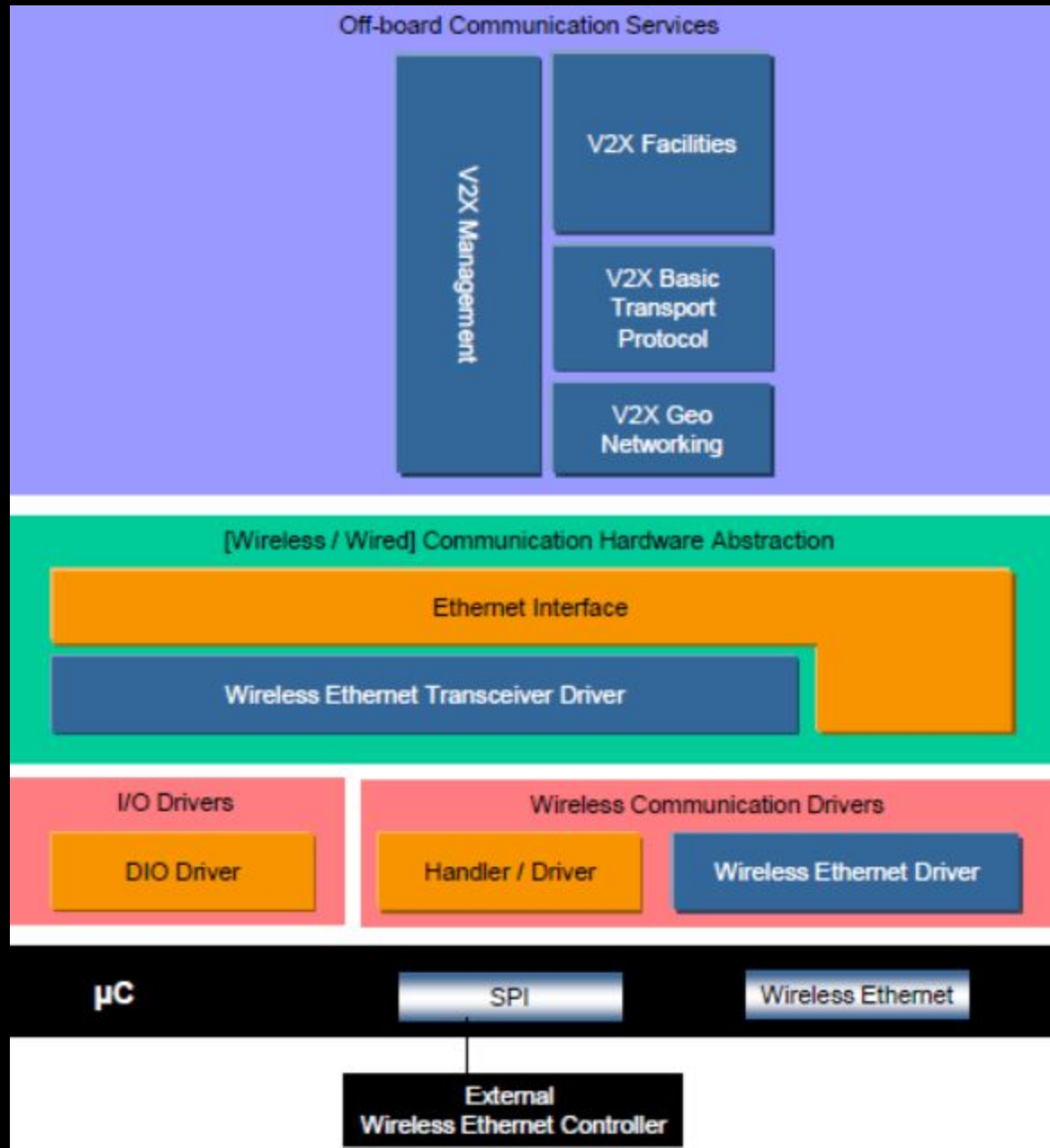


Note: This image is not complete with respect to all internal communication paths.

[1] The Interface between PduR and Tp differs significantly compared to the interface between PduR and the Ifs. In case of TP involvement a handshake mechanism is implemented allowing the transmission of I-Pdus > Frame size.
[2] CanIf with TTCAN serves both CanDrv with or without TTCAN. CanIf without TTCAN cannot serve CanDrv with TTCAN.

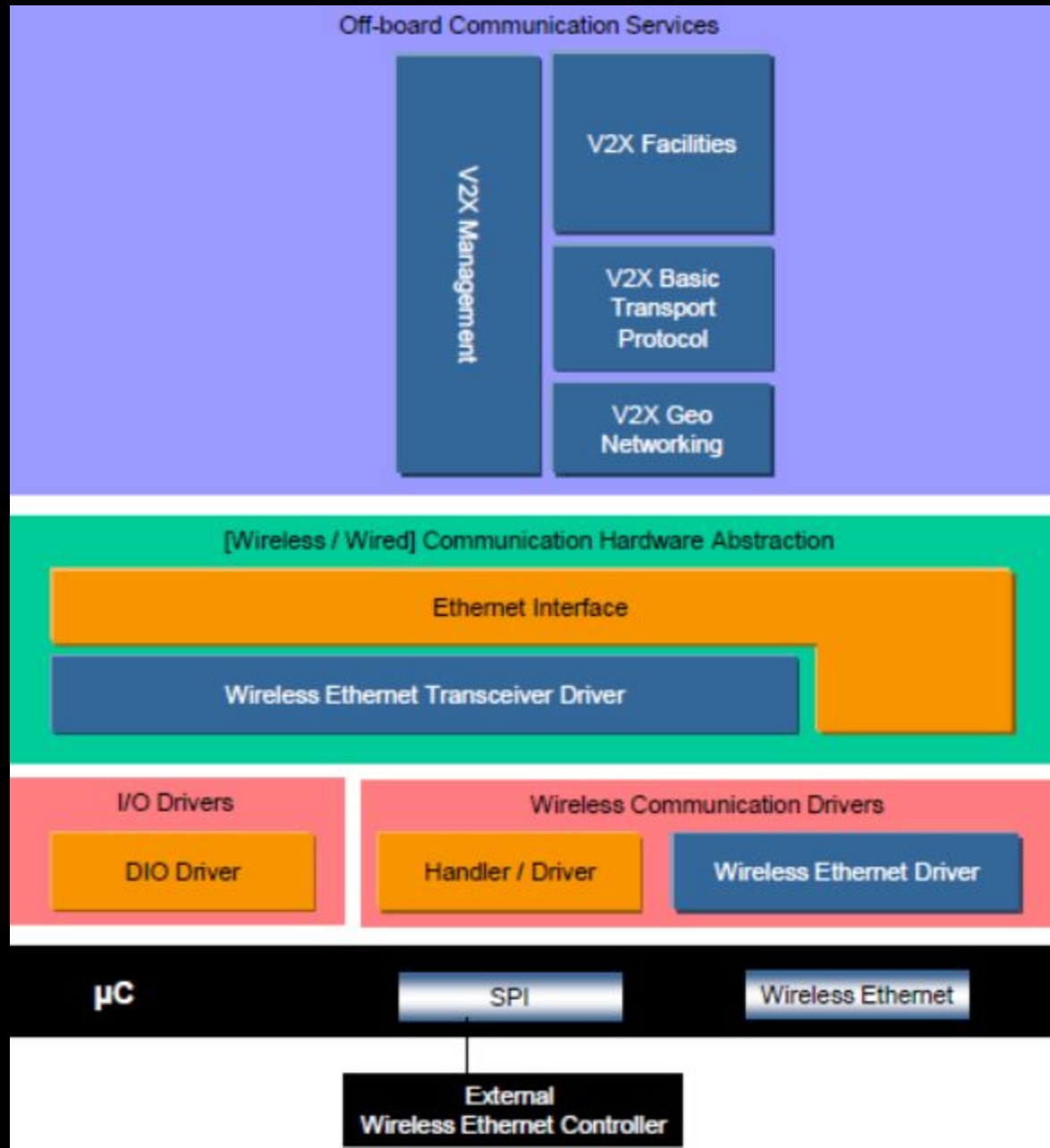
Off-board Communication Stack

- Group of modules for Vehicle-to-X communication via an ad-hoc wireless network.
- Provide a uniform interface to the Wireless Ethernet network which hide protocol and message properties from the application



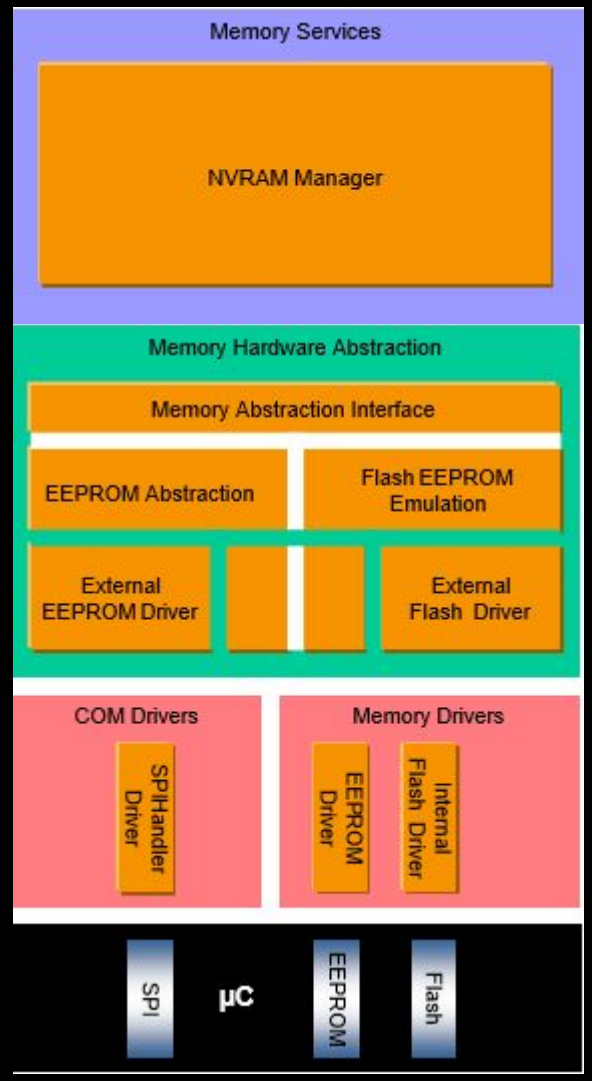
Off-board Communication Stack

- Facilities
  - implement the functionality for reception and transmission of standardized V2X messages
  - build the interface for vehicle specific SW-Cs
- Basic Transport Protocol (Layer 4)
- Geo-Networking (Layer 3)
  - Addressing based on geographic areas, the respective Ethernet frames have their own Ethernet Type.
- V2X Management
  - manages cross-layer functionality
  - like dynamic congestion control, security, position and time



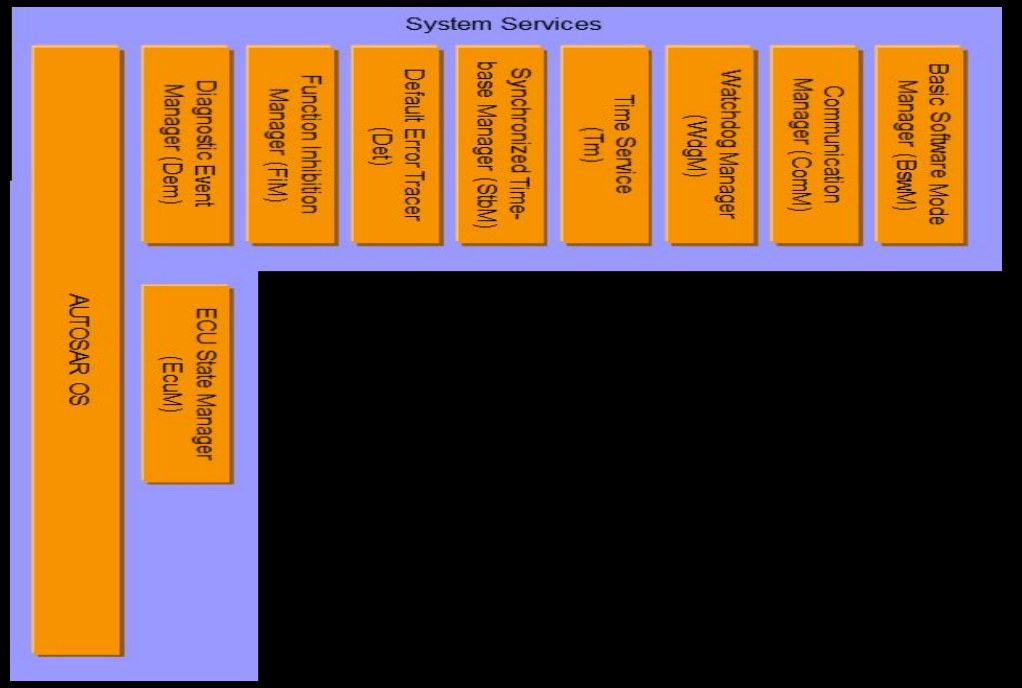
Memory Stack

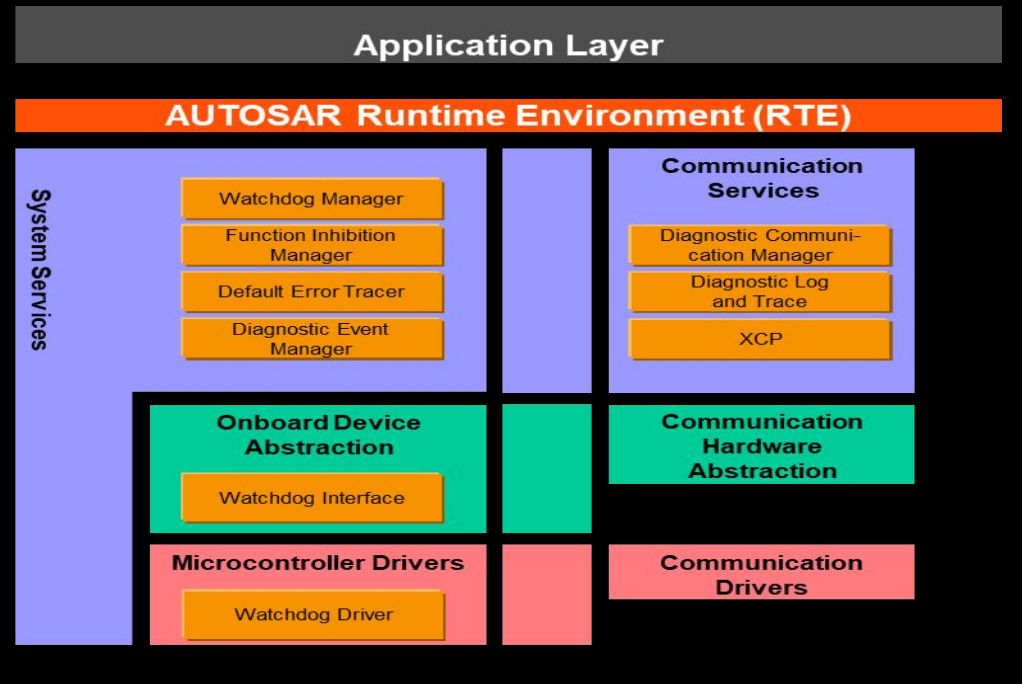
- Provide non volatile data to the application in a uniform way.
- Abstract from memory locations and properties.
- Provide mechanisms for non volatile data management like
  - Saving
  - Loading
  - Checksum protection and verification
  - Reliable storage etc.
- Fee Module
  - Allocate in ECUAL
  - Emulating an EEPROM abstraction on top of Flash hardware units a common access via Memory Abstraction Interface to both types of hardware is enabled.



System Stack

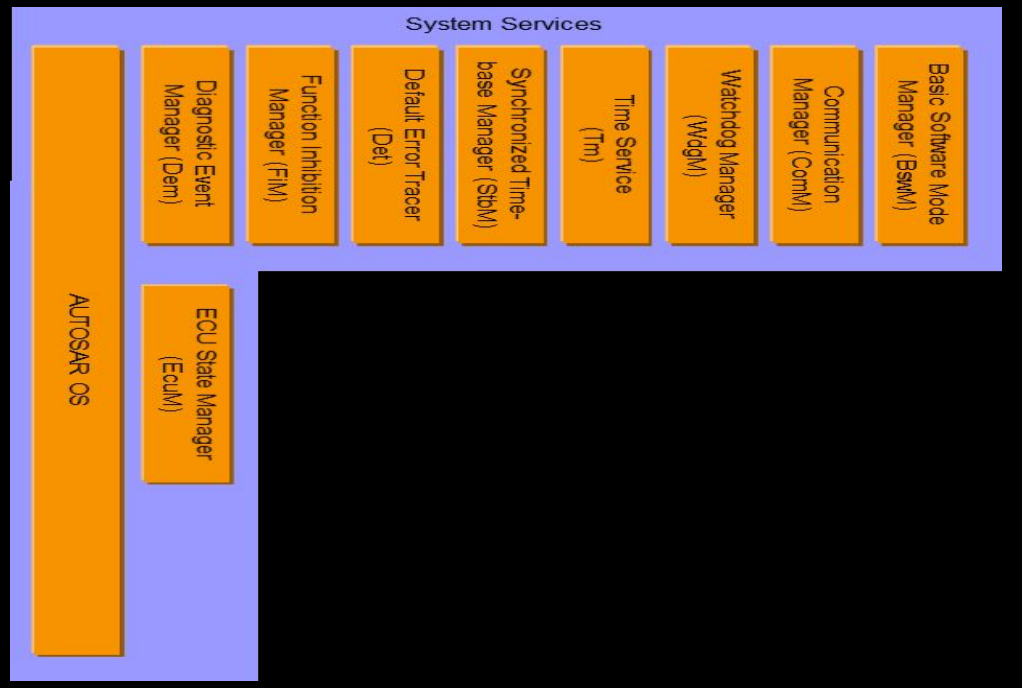
- Group of modules and functions which can be used by modules of all layers.
- Examples are Real Time Operating System (which includes timer services) and Error Manager.
- Some of these services are:
  - µC dependent (like OS)
  - Partly ECU hardware and application dependent (like ECU State Manager) or
  - Hardware and µC independent.

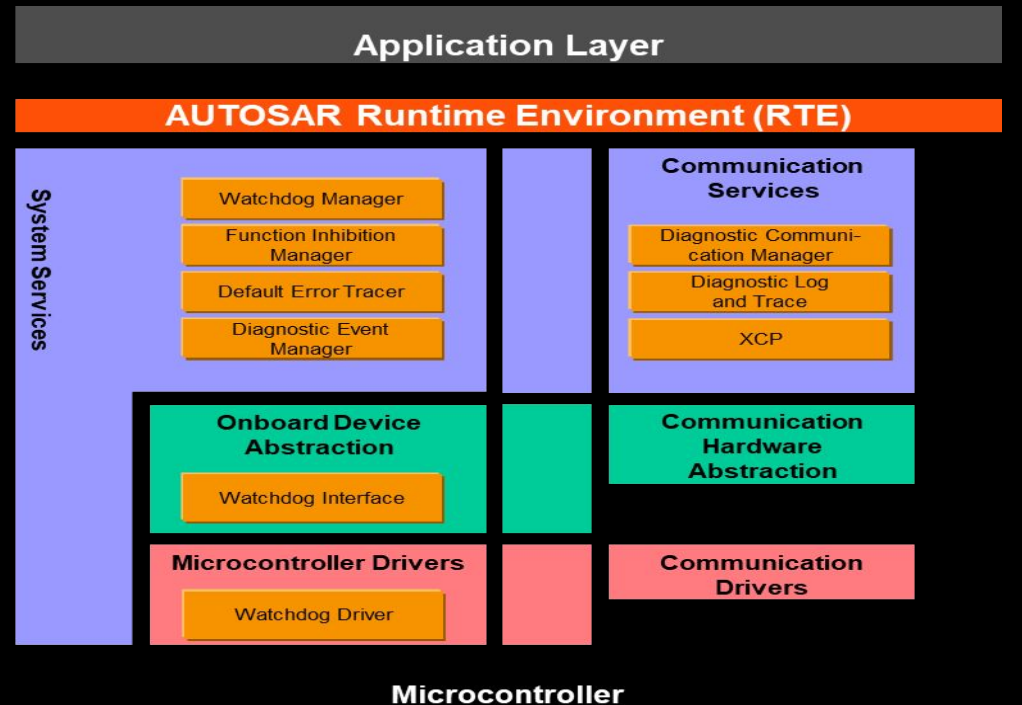




System Stack

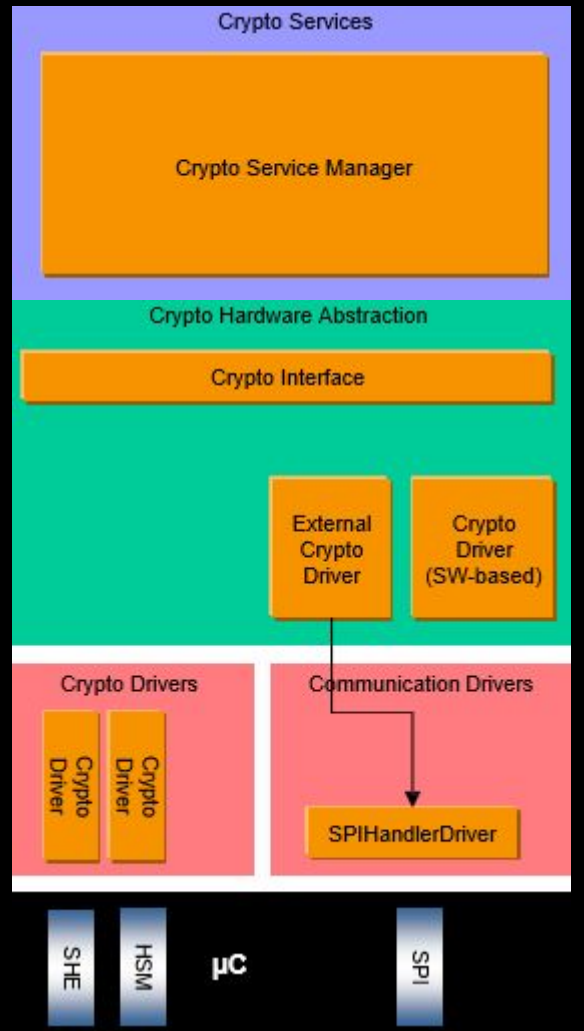
- Error handling has dedicated modules in AUTOSAR
  - Diagnostic Event Manager (DEM)
    - Responsible for processing and storing diagnostic errors (events) and associated data.
  - Diagnostic Log and Trace(DLT)
    - Supports logging and tracing of applications. It collects user defined log messages and converts them into a standardized format.
  - Default Error Tracer (DET)
    - All detected development errors in the Basic Software.





Crypto Stack

- Responsible for the management of cryptographic jobs and storage of keys.
- Provide cryptographic primitives and key storage to the application in a uniform way

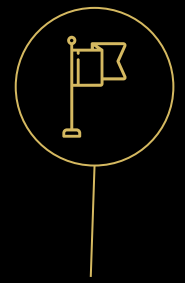


Course Content

- Automotive Software Industry
- Autosar Overview
- CAN Network
- Classic Autosar
  - Autosar Architecture
  - **Autosar Concepts and Methodology**
  - Autosar Application and RTE
  - Autosar Communication stack
  - Autosar IO stack
  - Autosar Memory Stack and RTE
  - Autosar OS and RTE
  - Autosar Watchdog Stack
  - Cybersecurity and Autosar Crypto Stack

Classic Autosar

Architecture



Methodology

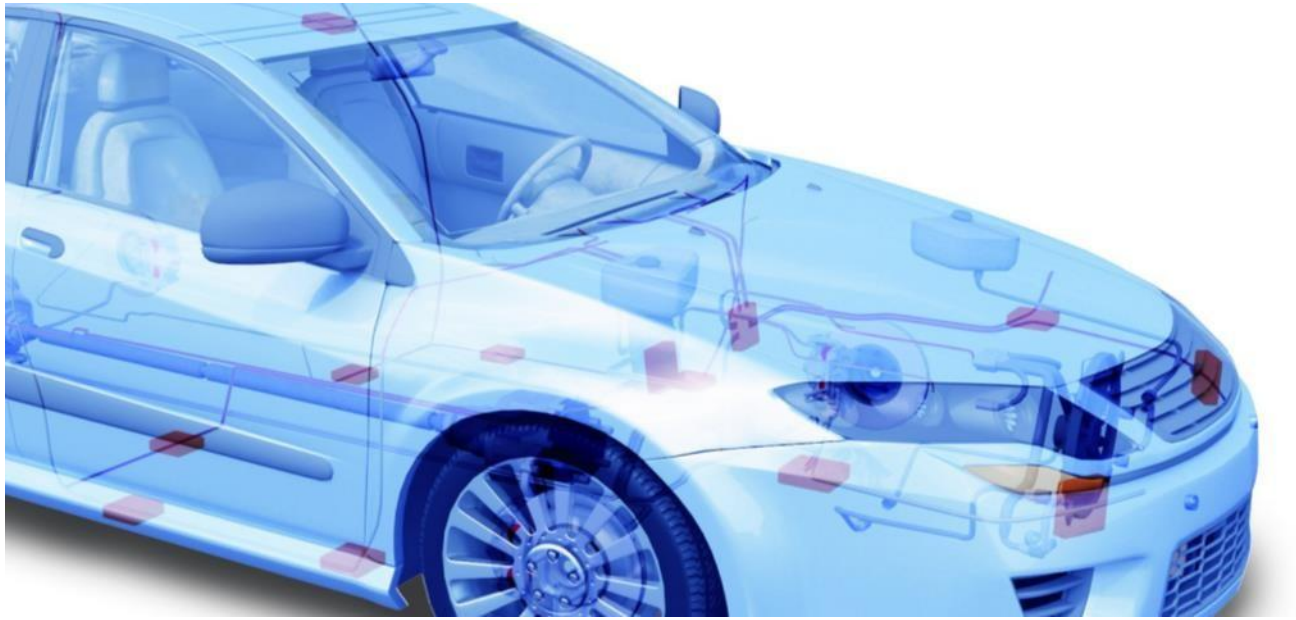
Concept

SwitchEvent
switch_event (event)
AUTOSAR Int.

LightRequest
switch_event(event)
request_light (type, mode)
AUTOSAR Interface

Front-Light Manager
request_light(type, mode)
set_light(type, mode)
AUTOSAR Interface

Xenonlight
set_light(type, mode)
set_current (…)
AUTOSAR Interface



SwitchEvent

check_switch ()

switch_event (event)

AUTOSAR Int.

LightRequest

switch_event(event)

request_light (type, mode)

AUTOSAR Interface

Front-Light Manager

request_light(type, mode)

get_keyposition()

set_light(type, mode)

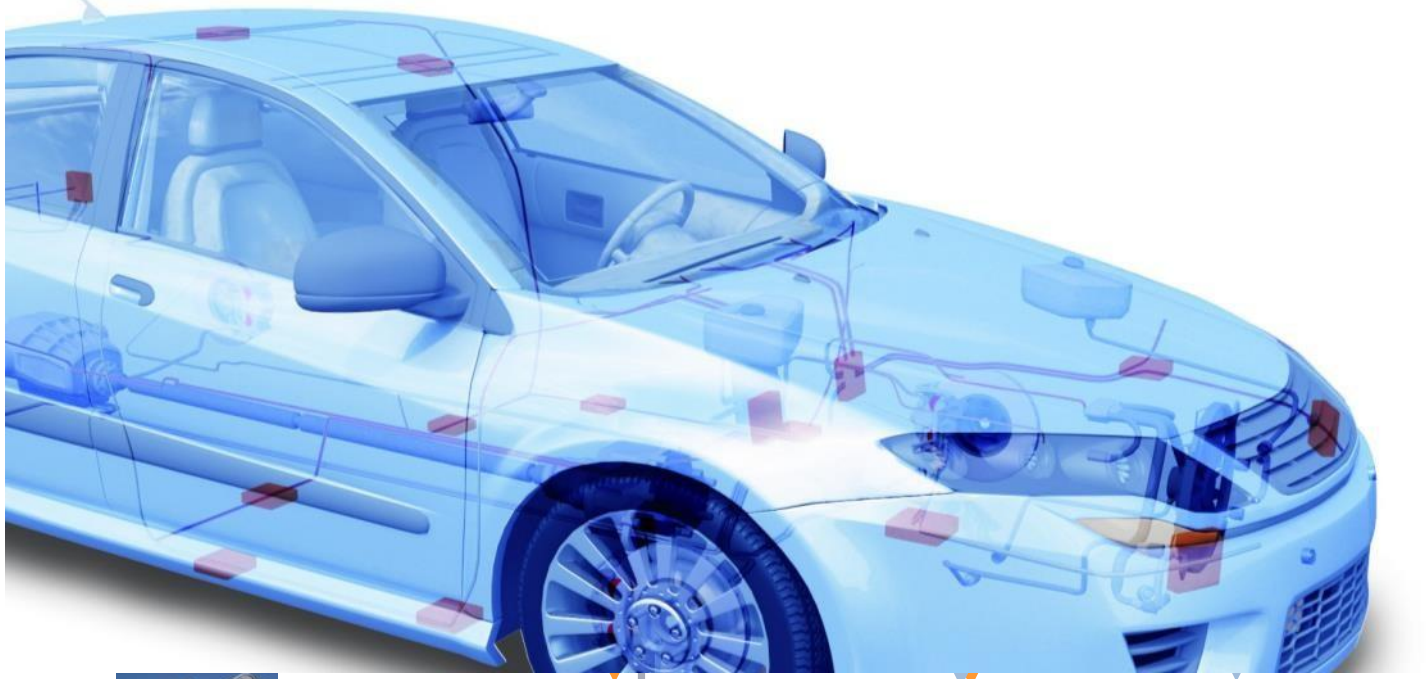
AUTOSAR Interface

Xenonlight

set_light(type, mode)

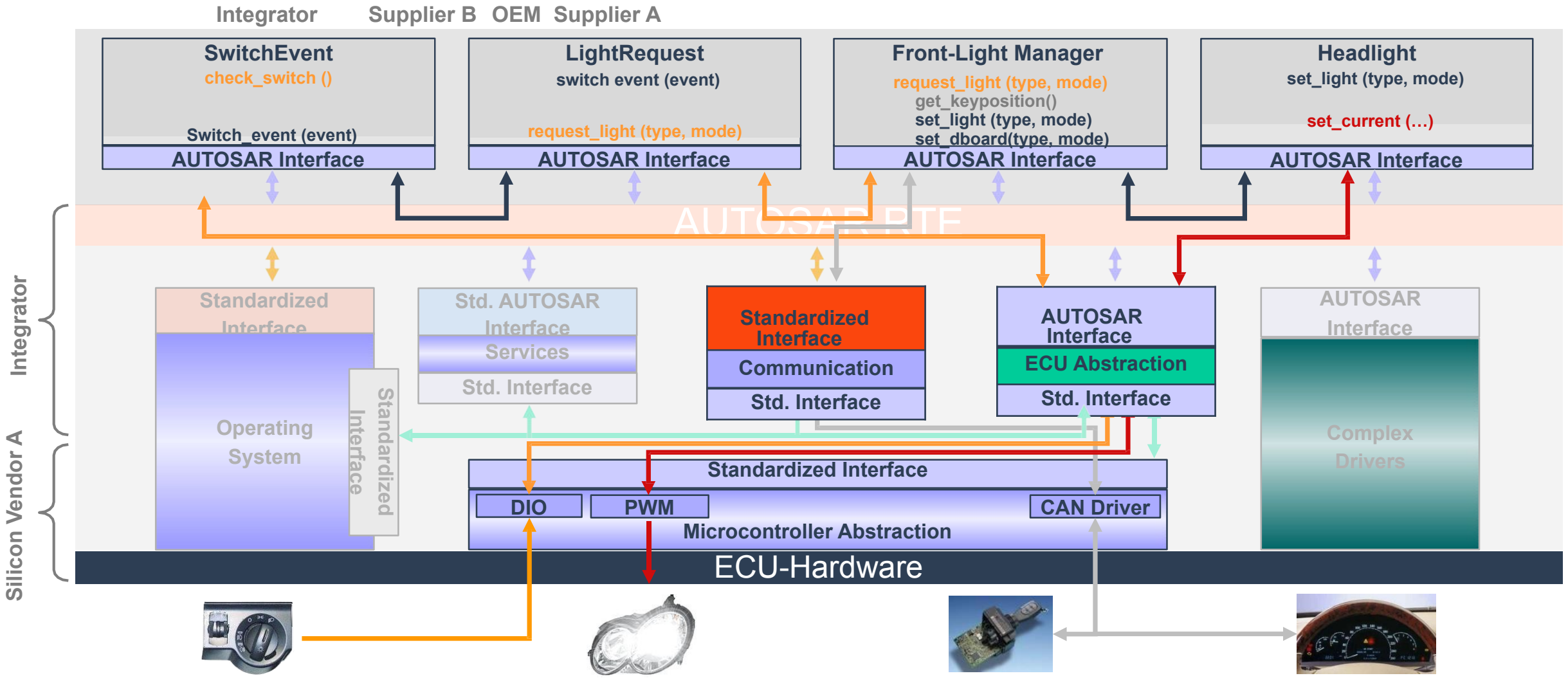
set_current (…)

AUTOSAR Interface

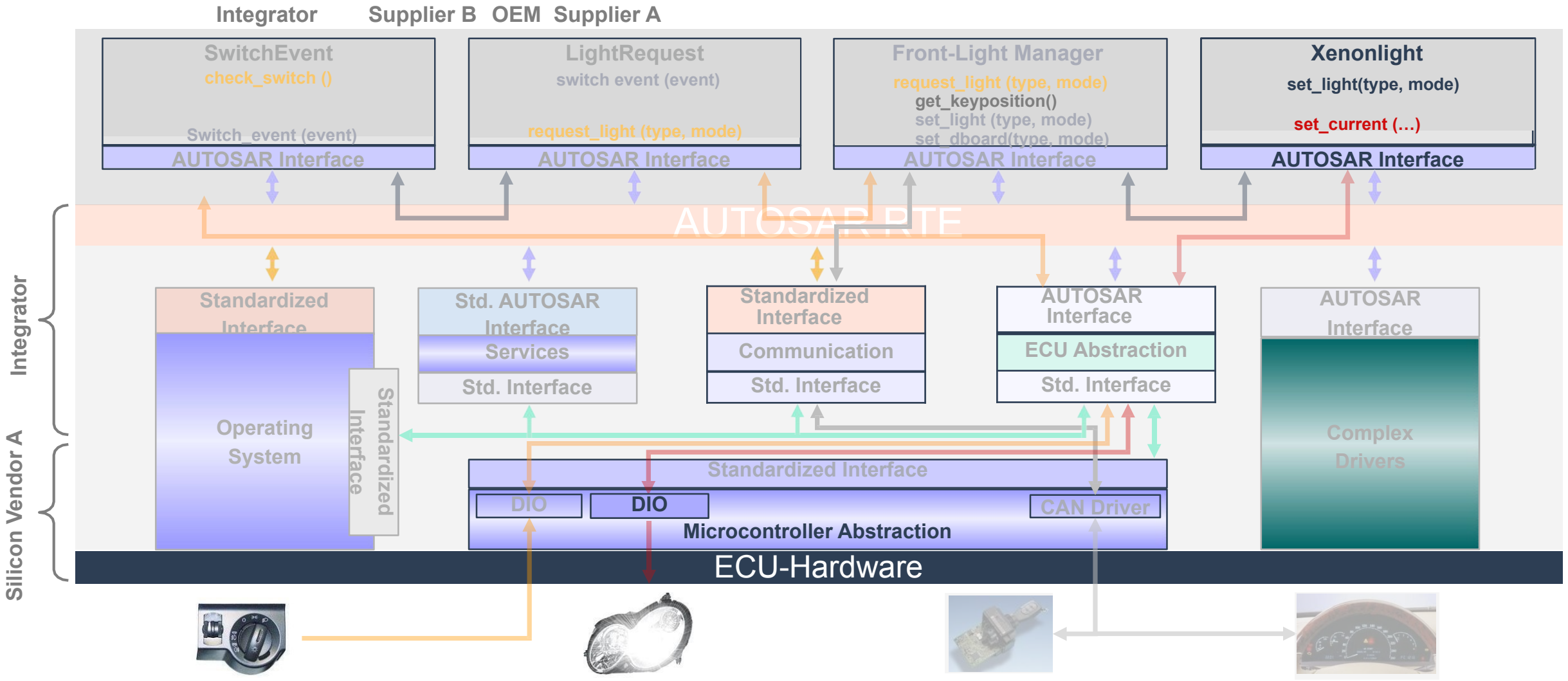


CAN Bus

Use Case ‘Front Light Management’: Exchange Type of Front Light

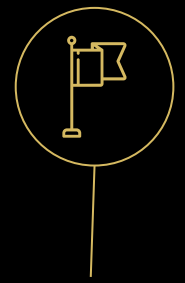


Use Case ‘Front Light Management’: Exchange Type of Front Light



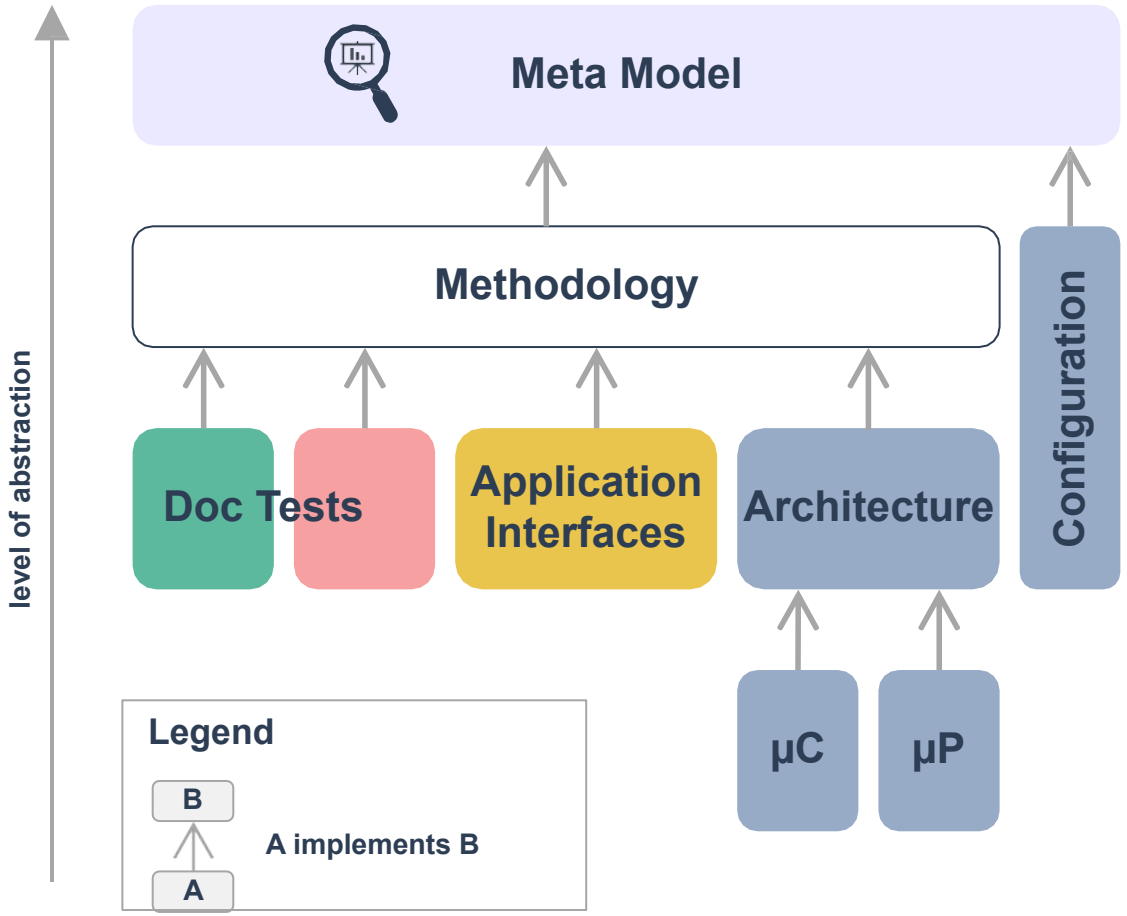
Classic Autosar

Architecture



Methodology

Concept



… provides means to describe the AUTOSAR **architecture** with all its **interfaces**

… defines **exchange formats** and description **templates** (e.g. manifest) to enable

- a seamless integration of the complete vehicle E/E architecture,
- the automatized configuration of the µC- and µP-software stacks and
- the seamless integration of application software

… supports means to **ensure safety** and **security** of the system

… provides templates to **document the standard**

Course Content

- Automotive Software Industry
- Autosar Overview
- CAN Network
- Classic Autosar
  - Autosar Architecture
  - Autosar Concepts and Methodology
  - **Autosar Application and RTE**
  - Autosar Communication stack
  - Autosar IO stack
  - Autosar Memory Stack and RTE
  - Autosar OS and RTE
  - Autosar Watchdog Stack
  - Cybersecurity and Autosar Crypto Stack

RTE

- **Autosar Application and RTE**

https://www.btc-es.de/en/blog/autosar-what-every-function-developer-should-know.html