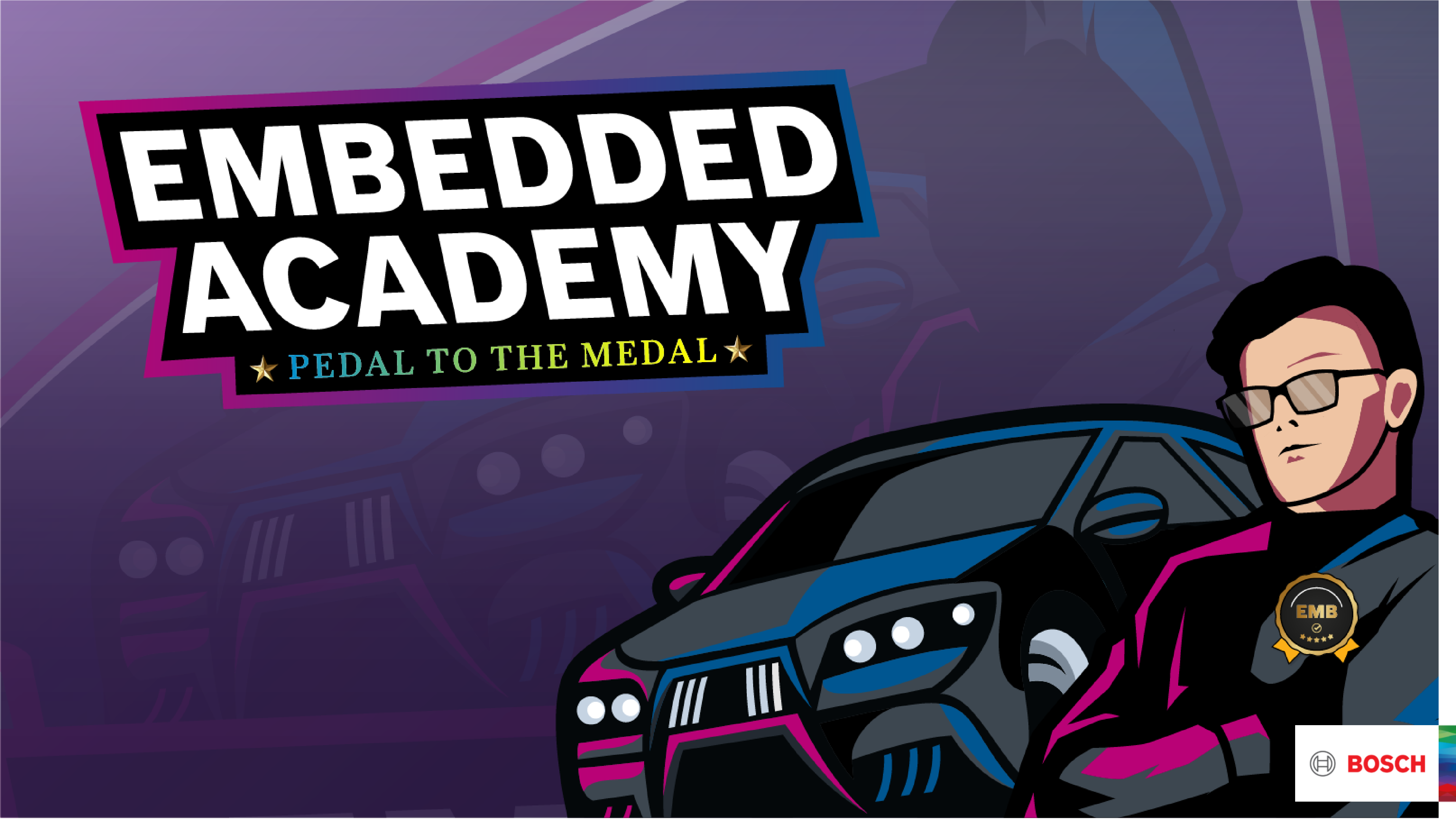

EMBEDDED ACADEMY
PEDAL TO THE MEDAL
EMB
BOSCH

BGSV Embedded Academy (BEA)

Focused Program to Develop Embedded Competence

BGSV EMBEDDED ACADEMY

Technical Competence

- T1: Automotive Basics (Sensor, SW, Mobility Solution)
- T2: Automotive SW Architecture (AUTOSAR)
- T3: Embedded Programming
- T5: Test Overview

Methodological Competence

- M1: SW Development Lifecycle
- M3: Clean Code

Process Competence

- P1: Requirements Engineering
- P2: Design Principles
- P3: Review
- P4: Safety & Security

Classroom training, Online Self-learning, Live Demo

Purpose: Develop basic general embedded competence

Disclaimer

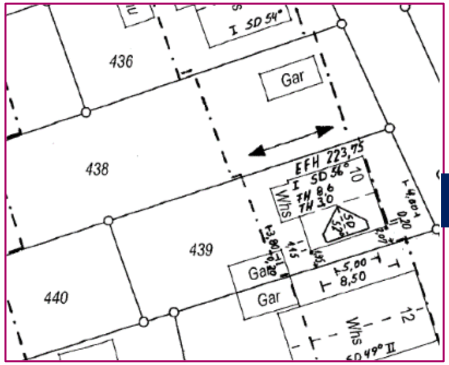
- This slide is a part of BGSV Embedded Academy (BEA) program and only used for BEA training purposes.
- This slide is Bosch Global Software Technology Company Limited's internal property.
- This slide has some copyright images and text, which belong to the respective organizations.

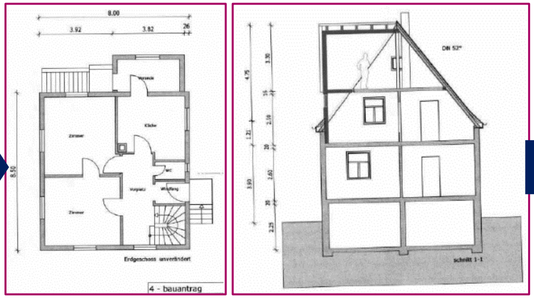
T2
AUTOMOTIVE SOFTWARE ARCHITECTURE

WHAT IS SOFTWARE ARCHITECTURE?

What is Software Architecture
Build a house

- Surveyor plan
- Floor plan
- 2D or 3D views
- Implementation





Plot environment (adjacent to neighbors‘ plots), permissible and used building footprint

Position and size of walls, wall openings, floors and ceilings

3D representation of the building or parts of it. Location and views of furniture and fixtures. Virtual tour inside the object.

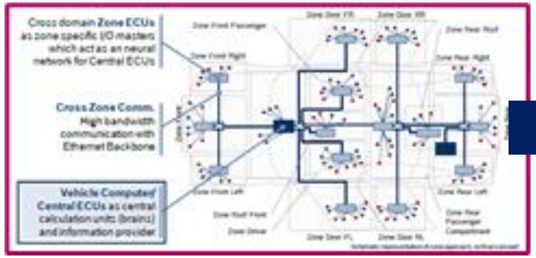
Physical realization: built home (ready to move in).

Bosch Global Software Technologies
alt_future

What is Software Architecture

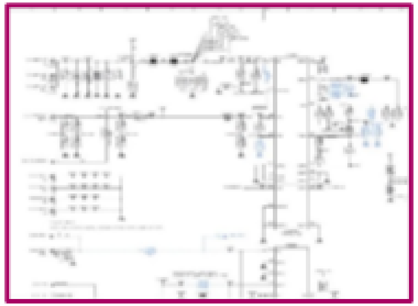
How to build an ECU

▶ E/E architecture



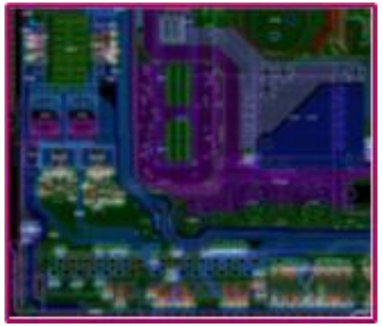
Network and electrical environment of the ECU

▶ Circuit



Logical wiring of components describing electrical functions

▶ PCB Layout

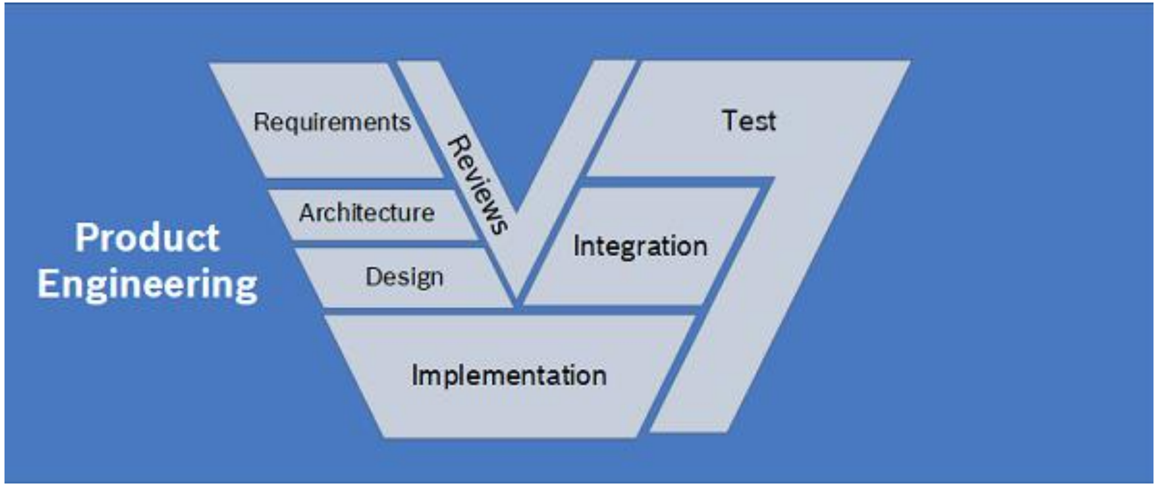


Physical location and wiring of components (considering of non-functional requirements like EMV, heat dissipation, physical interfaces to housing)

▶ Implementation

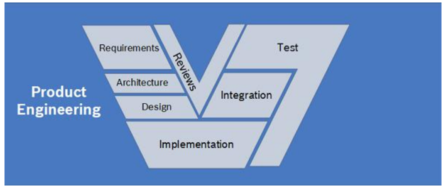
Physical realization: anufactured Electronic Control Unit to be installed into the vehicle.

What is Software Architecture
How to build a software product – Oversimplify



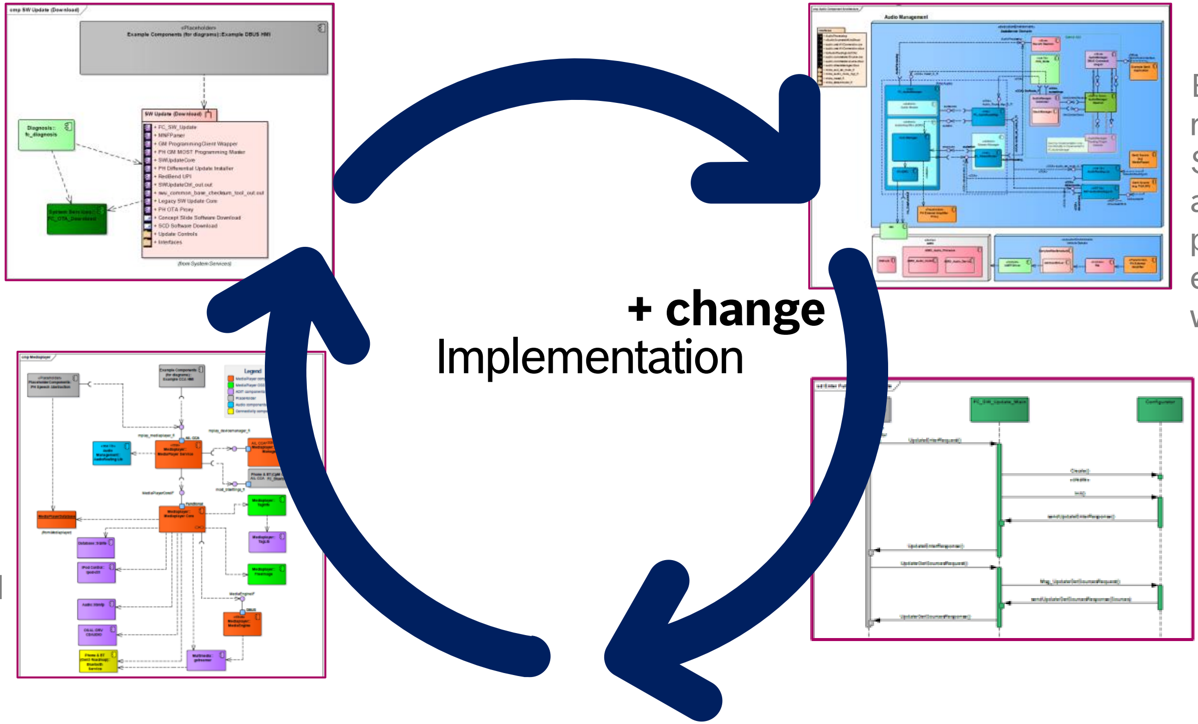
Bosch Global Software Technologies
alt_future

What is Software Architecture
How to build a software product – Architectural Views



- **Context view**
  Embedding of SW systems (as black box) in its environment, interfaces to neighboring systems (e.g. through different communication channels) (Relation with new E/E architectures).

- **Component view (+interfaces)**
  Static (hierarchical) composition of the SW system consisting of architectural building blocks, subsystems, SW components and their interfaces.



- **Deployment view**
  Environment in which the SW is running: HW components running the SW, processors, network topologies and protocols, as well as further physical components of the system environment. The component view within the environment is optional.

- **Dynamic runtime view**
  Description of run time behavior of existing SW elements and their concurrence. Dynamical structures.

What is Software Architecture
What is an architecture

Definition of Architecture (IEEE, 2011-12-01):

> ***architecture*** *<system> fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*

According to this definition architectures describe:

- fundamental **concepts** on which corresponding systems are built,
- the **environment** where the system under design need to be integrated into,
- **components** which the system consists of, and
- **relations** between the components and the environment.

What is Software Architecture
Quiz time

- Select the **three most often used** architecture views:

(a) Physical database view

(b) Context view

(c) Building Block/Component view

(d) Test-driven view

(e) Configuration view

(f) Runtime view

What is Software Architecture
Quiz time

- Select the **three most often used** architecture views:

(a) Physical database view

(b) Context view

(c) Building Block/Component view

(d) Test-driven view

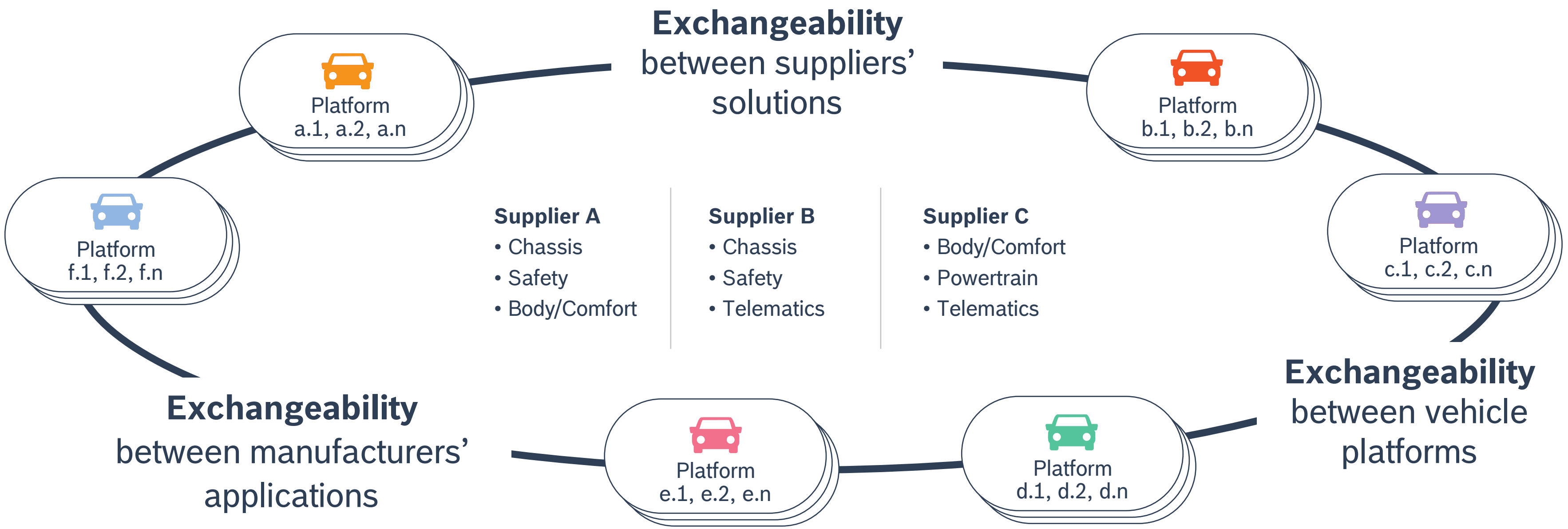
(e) Configuration view

(f) Runtime view

WHAT IS AUTOSAR?

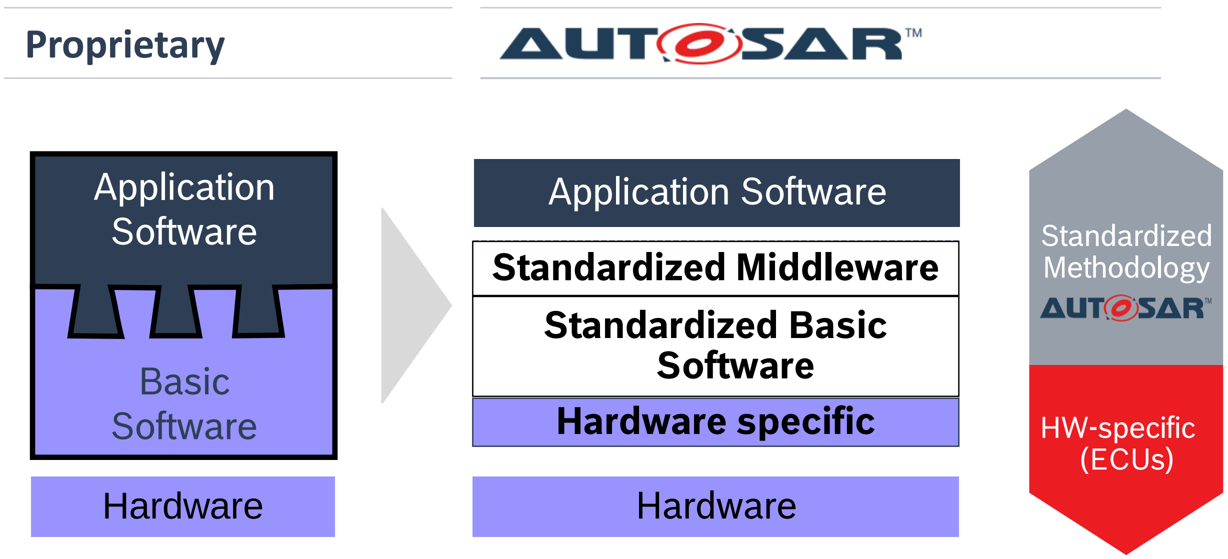
What is AUTOSAR
AUTOSAR Vision

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



What is AUTOSAR
Aims and benefits of using AUTOSAR

- AUTOSAR aims to standardize the software architecture of **E**lectronic **C**ontrol **U**nits (**ECUs**). AUTOSAR paves the way for innovative electronic systems that further improve performance, safety and security.



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

What is AUTOSAR
More Than 280 AUTOSAR Partners

9 Core Partners

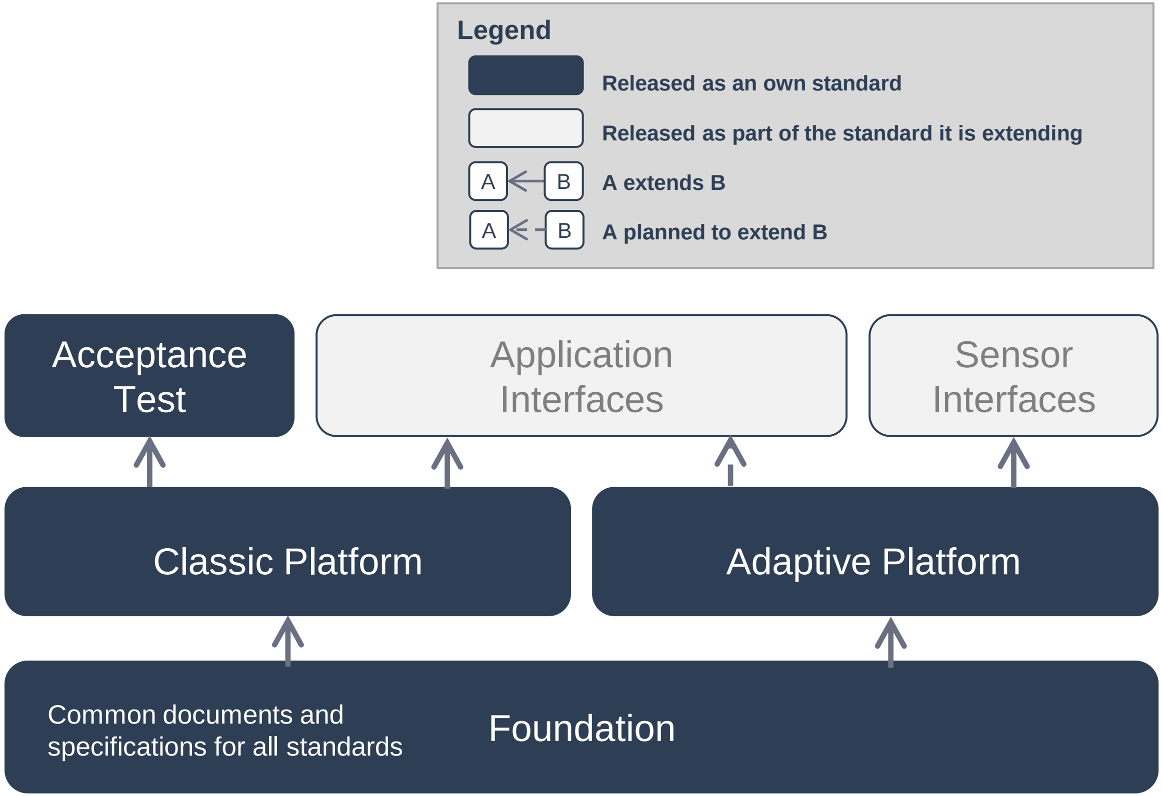
58 Premium Partners

2 Strategic Partners

53 Development Partners

+ 152 Associate Partners
+ 29 Attendees

What is AUTOSAR
AUTOSAR Deliverables



Most common type of deliverables

- ATS: Acceptance Test Specification
- CONC: Concept document
- EXP: Explanation document
- MMOD: Meta-model files (M2)
- MOD: Model files (M1)
- PRS: Protocol Specification
- RS/SRS: Requirement Specification
- SWS: Software Specification
- TPS: Template Specification
- TR: Technical Report

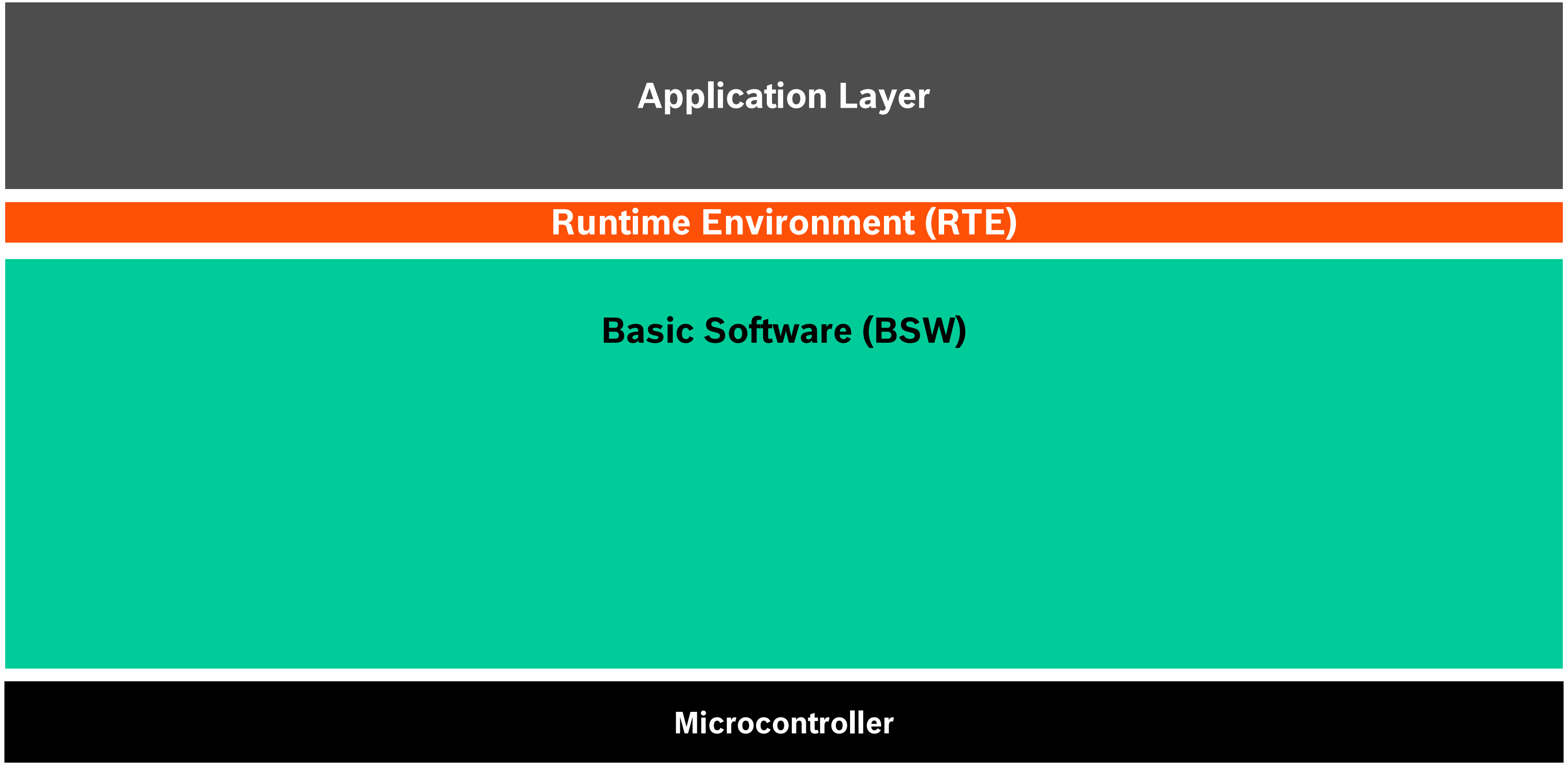
AUTOSAR SVN copy @Bosch:
file://///si8256.de.bosch.com/AUTOSAR$/SVN3-COPY/26_Standards/02_Releases/

AUTOSAR Docupedia @Bosch:
https://inside-docupedia.bosch.com/confluence/display/AUT

AUTOSAR LAYERED Architecture

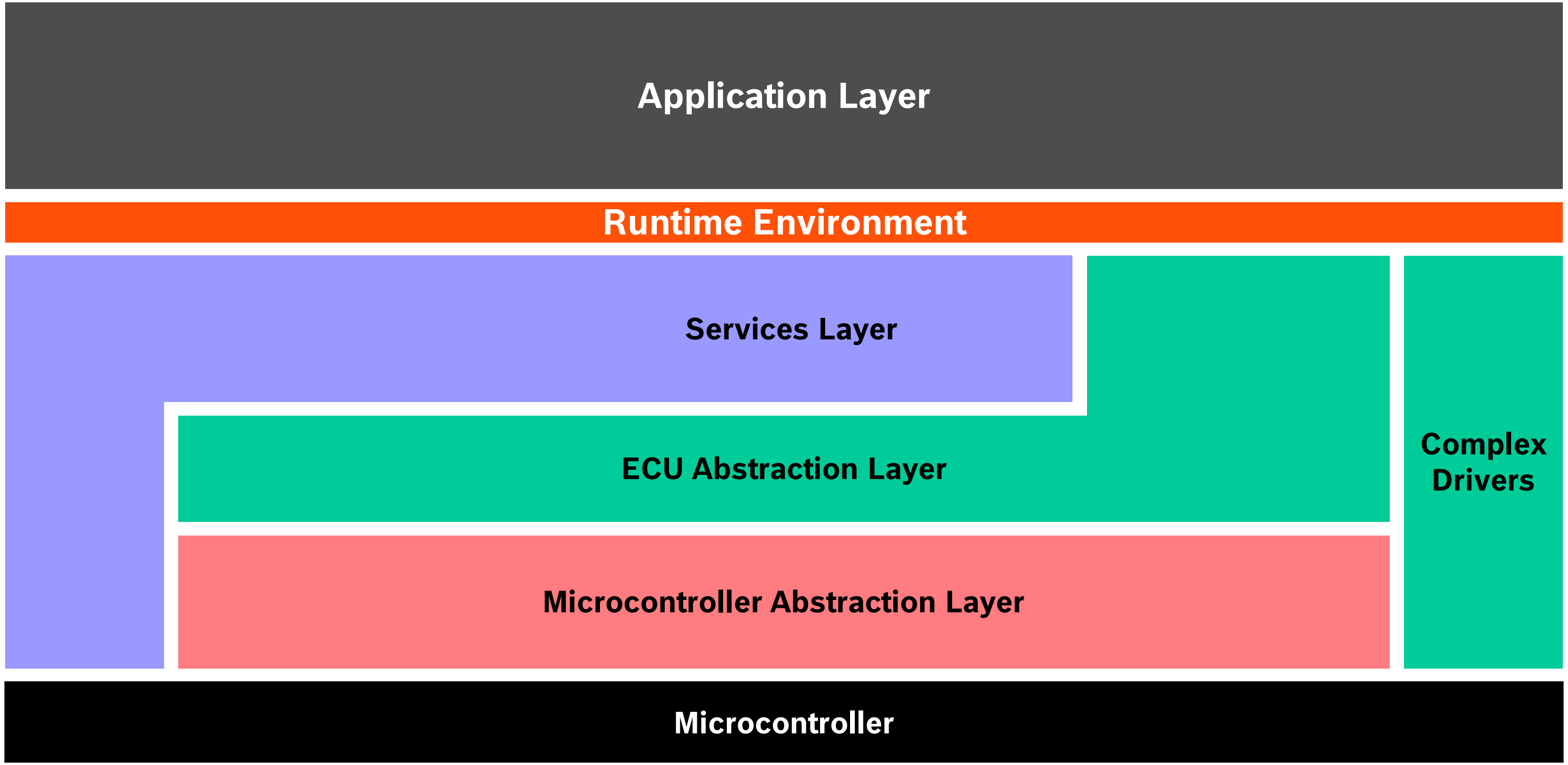
AUTOSAR Layered Architecture
Top View

The AUTOSAR Architecture distinguishes on the highest abstraction level between three software layers: Application, Runtime Environment and Basic Software which run on a Microcontroller.



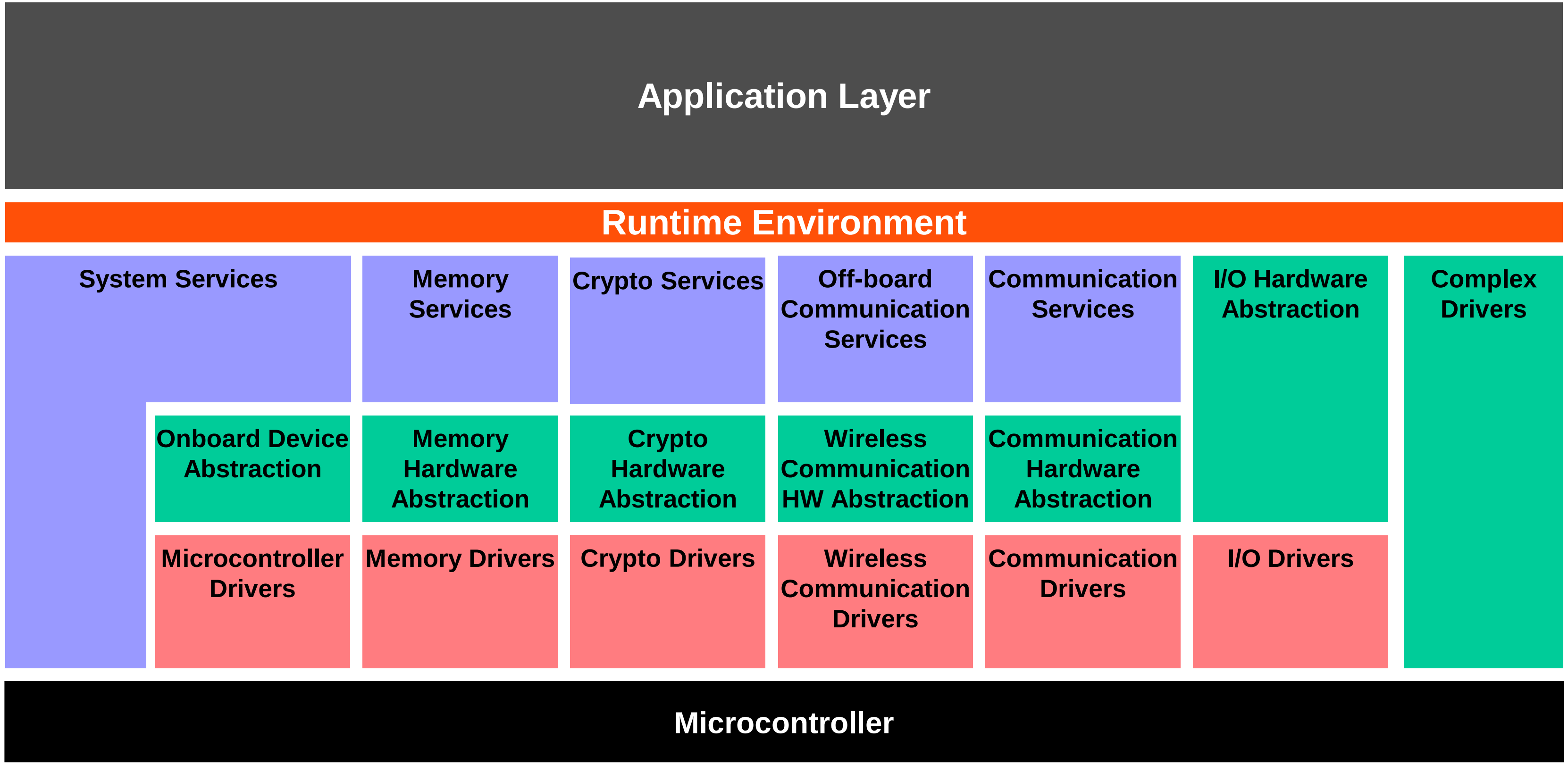
AUTOSAR Layered Architecture
Coarse view

The AUTOSAR Basic Software is further divided in the layers: Services, ECU Abstraction, Microcontroller Abstraction and Complex Drivers.



AUTOSAR Layered Architecture
Detailed view

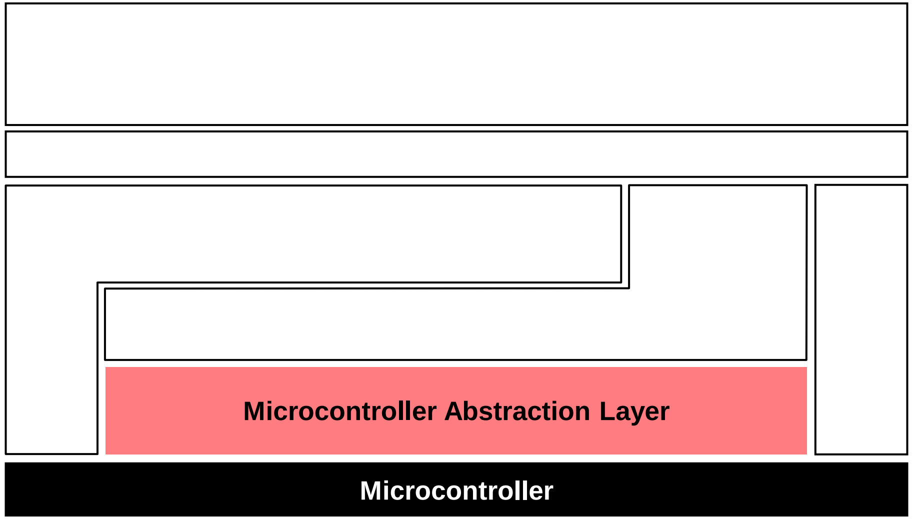
The Basic Software Layers are further divided into functional groups. Examples of Services are System, Memory and Communication Services.



AUTOSAR Layered Architecture
Microcontroller Abstraction Layer

The **Microcontroller Abstraction Layer** 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

AUTOSAR Layered Architecture
ECU Abstraction Layer

The **ECU Abstraction Layer** interfaces the drivers of the Microcontroller Abstraction Layer. It also contains drivers for external devices.

It offers an API for access to peripherals and devices regardless of their location (µC internal/external) and their connection to the µC (port pins, type of interface)

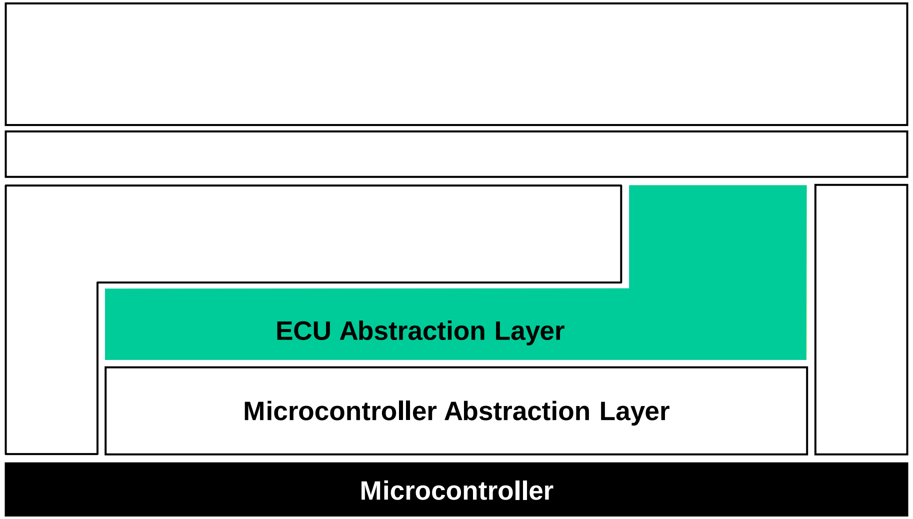
Task

Make higher software layers independent of ECU hardware layout

Properties

Implementation: µC independent, ECU hardware dependent

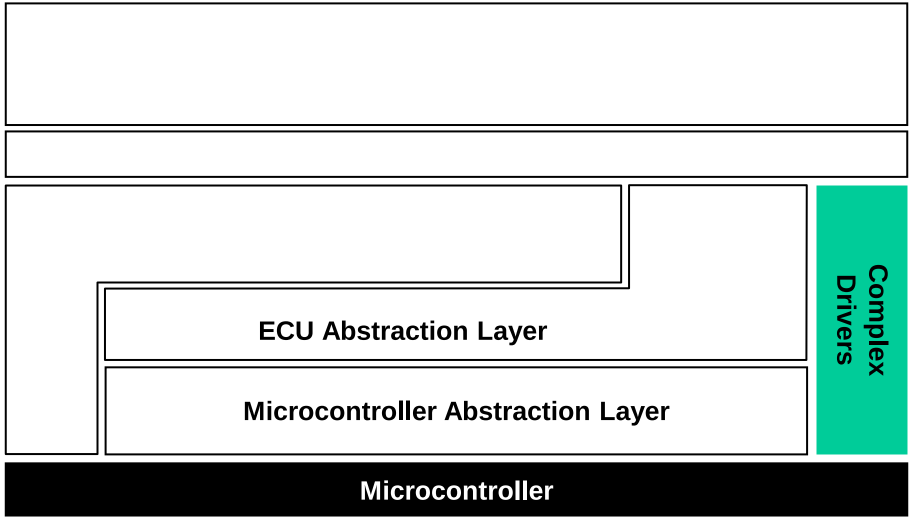
Upper Interface: µC and ECU hardware independent



Bosch Global Software Technologies
alt_future

AUTOSAR Layered Architecture
Complex Drivers

The **Complex Drivers Layer** spans from the hardware to the RTE.



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.

Properties

Implementation: might be application, µC and ECU hardware dependent

Upper Interface: might be application, µC and ECU hardware dependent

AUTOSAR Layered Architecture
Services Layer

The **Services Layer** is 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)

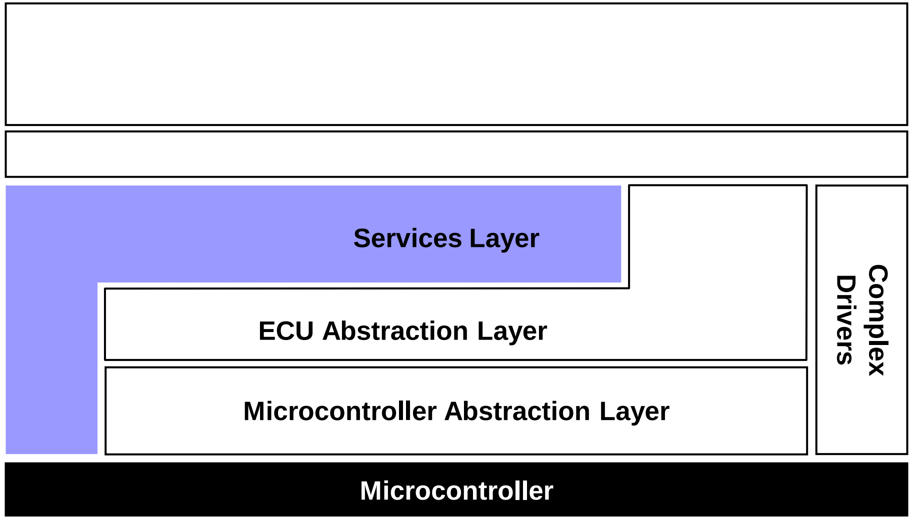
Task

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



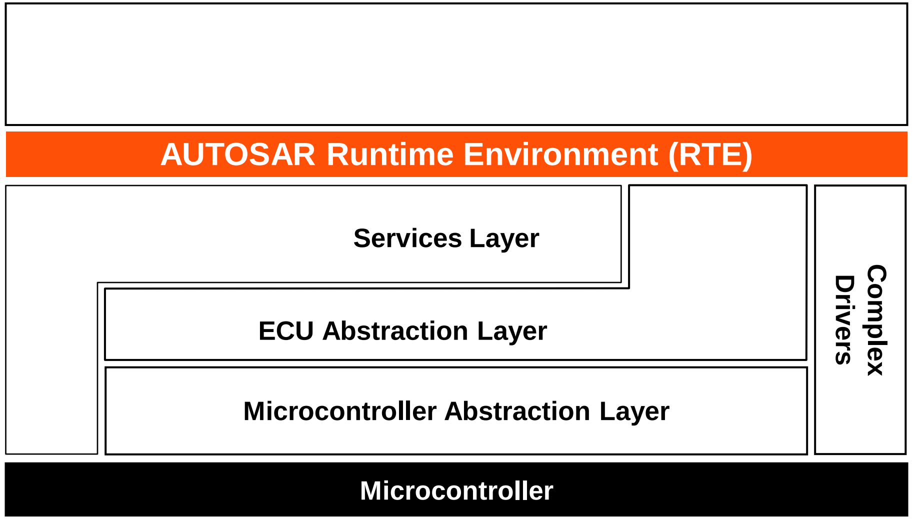
Bosch Global Software Technologies
alt_future

AUTOSAR Layered Architecture
AUTOSAR Runtime Environment (RTE)

The **RTE** is a layer providing communication services to the application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components).

Above the RTE the software architecture style changes from “layered“ to “component style“.

The AUTOSAR Software Components 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 Basic Software
Quiz time

- **From the top to bottom, how many software layers of the highest abstraction level of AUTOSAR architecture?**

(a) 3 layers: Application, RTE, BSW.

(b) 5 layers: Application, RTE, Service layer, ECU Abstraction layer, MCAL.

(c) 3 layers: Services layer, Abstraction layer, MCAL.

(d) 4 layers: Application, RTE, BSW, MCAL.

RTE = Runtime Environment

BSW = Basic Software

MCAL = Microcontroller Abstraction Layer

AUTOSAR Basic Software

Quiz time

- **From the top to bottom, how many software layers of the highest abstraction level of AUTOSAR architecture?**

(a) 3 layers: Application, RTE, BSW.

(b) 5 layers: Application, RTE, Service layer, ECU Abstraction layer, MCAL.

(c) 3 layers: Services layer, Abstraction layer, MCAL.

(d) 4 layers: Application, RTE, BSW, MCAL.

RTE = Runtime Environment

BSW = Basic Software

MCAL = Microcontroller Abstraction Layer

AUTOSAR Layered Architecture
Quiz time

- **Which of the following qualities can most likely be improved by using a layered architecture?**

(a) Runtime efficiency (performance).

(b) Flexibility in modifying or changing the system.

(c) Flexibility at runtime (configurability).

(c) Non-repudiability.

AUTOSAR Layered Architecture
Quiz time

- **Which of the following qualities can most likely be improved by using a layered architecture?**

(a) Runtime efficiency (performance).

(b) Flexibility in modifying or changing the system.

(c) Flexibility at runtime (configurability).

(c) Non-repudiability.

AUTOSAR BASIC SOFTWARE

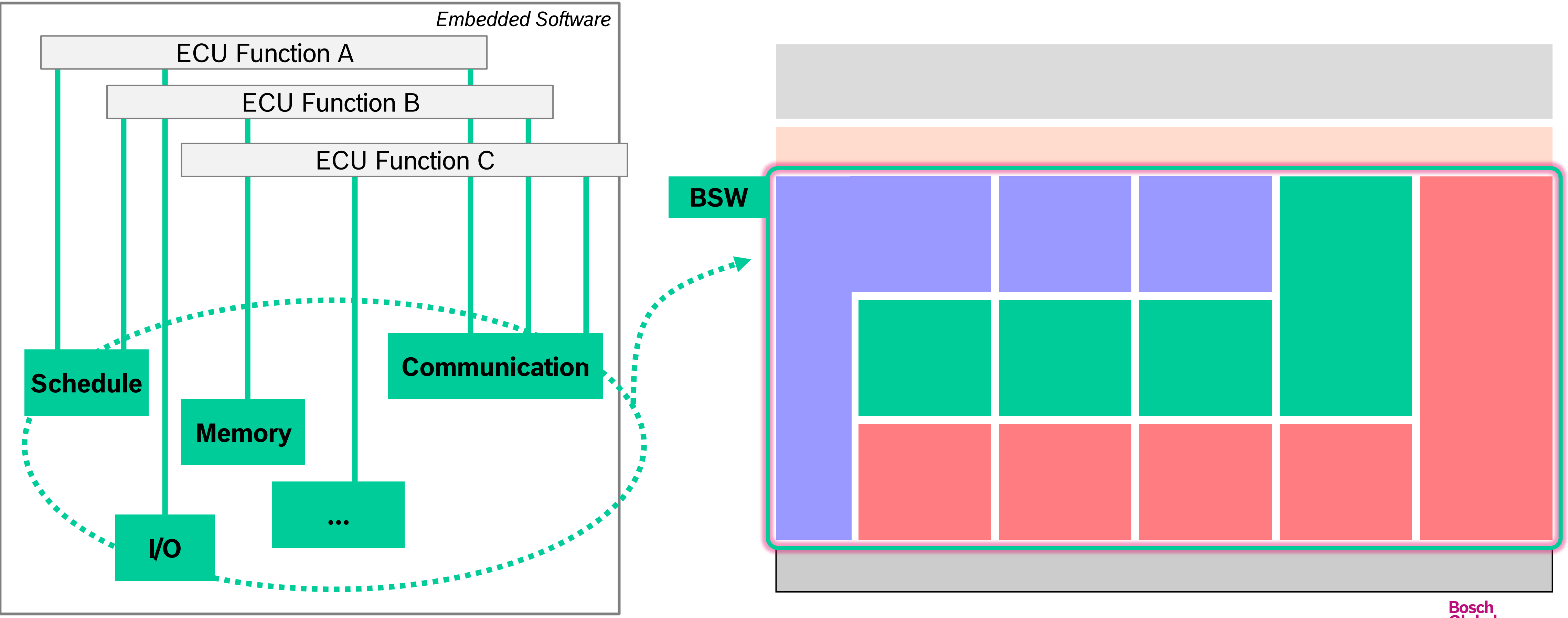
AUTOSAR Basic Software
What is “Basic SW”?

Basic Software

/ˈbeɪ.sɪk/, adj
simple and not complicated, so able to provide the base or starting point from which something can develop [cambridge.org]

/ˈsɑːft.wer/, noun
the instructions that control what a computer does; computer programs [cambridge.org]

AUTOSAR Basic Software
What is "basic" to Embedded Software?



AUTOSAR Basic Software

REF

Types of BSW modules

BSW Module

On-chip Driver

A driver contains the functionality to control and access an internal or an external device.

Internal devices are located inside the microcontroller. Examples for internal devices are:
- Internal EEPROM
- Internal CAN controller
- Internal ADC
A driver for an internal device is called internal driver and is located in the Microcontroller Abstraction Layer.

Off-chip Driver

External devices are located on the ECU hardware outside the microcontroller.

Examples for external devices are:
- External EEPROM
- External watchdog
- External flash
*A driver for an external device is called **external driver** and is located in the ECU Abstraction Layer. It accesses the external device via drivers of the Microcontroller Abstraction Layer.*
This way also components integrated in System Basis Chips (SBCs) like transceivers and watchdogs are supported by AUTOSAR.
Example: a driver for an external EEPROM with SPI interface accesses the external EEPROM via the handler/driver for the SPI bus.

Interface

An Interface (interface module) 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.
In general, interfaces are located in the ECU Abstraction Layer.
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).

Handler

A **handler** 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, ADC Driver).

Manager

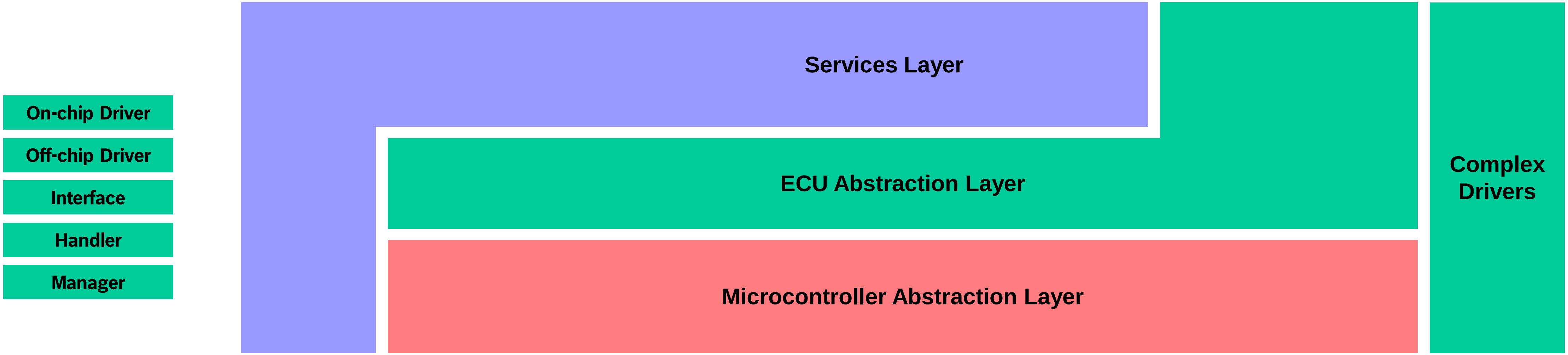
A **manager** offers specific services for multiple clients. It is needed in all cases where pure handler functionality is not enough to abstract from multiple clients.

Besides handler functionality, a manager can evaluate and change or adapt the content of the data.
*In general, managers are located in the **Services Layer***
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.

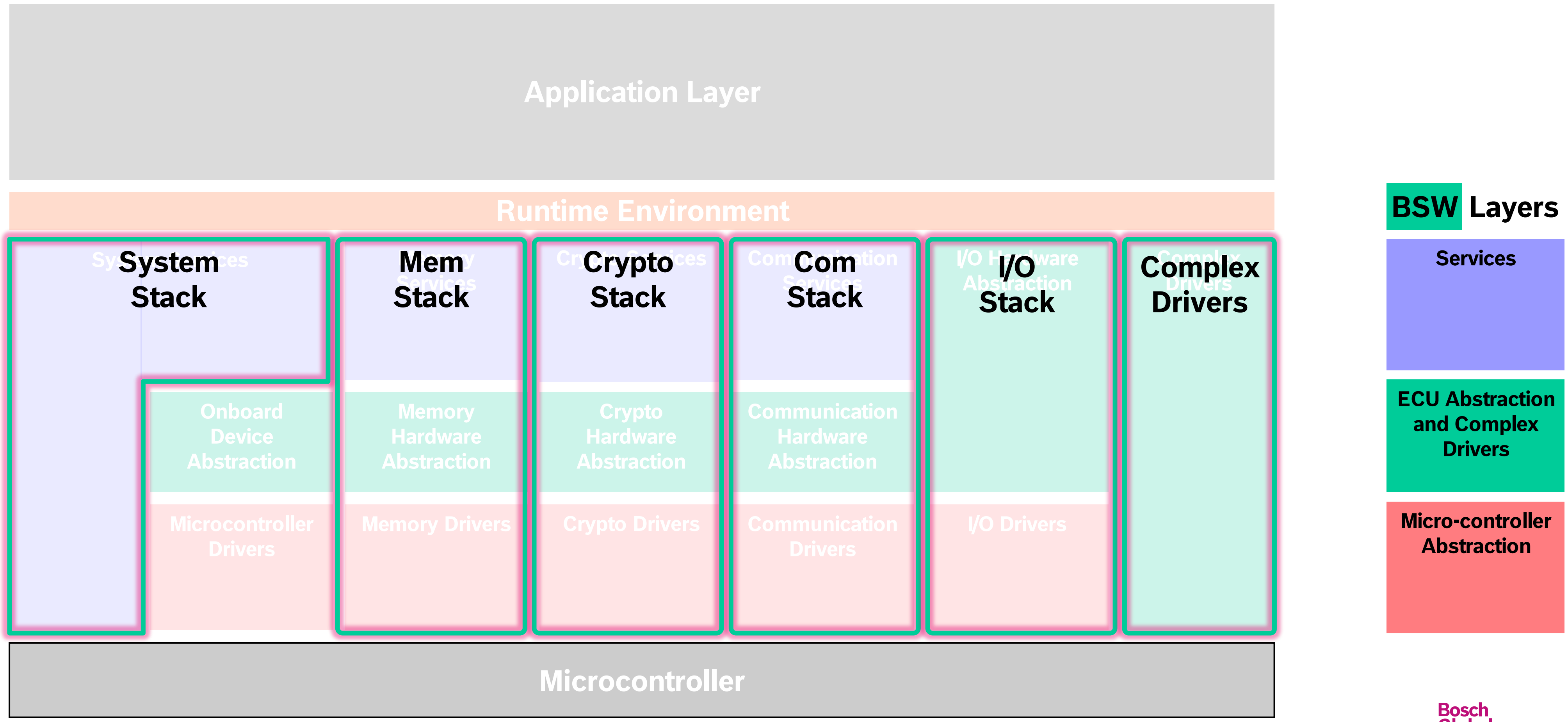
AUTOSAR Basic Software

Question 1

Map BSW module type to the suitable layer

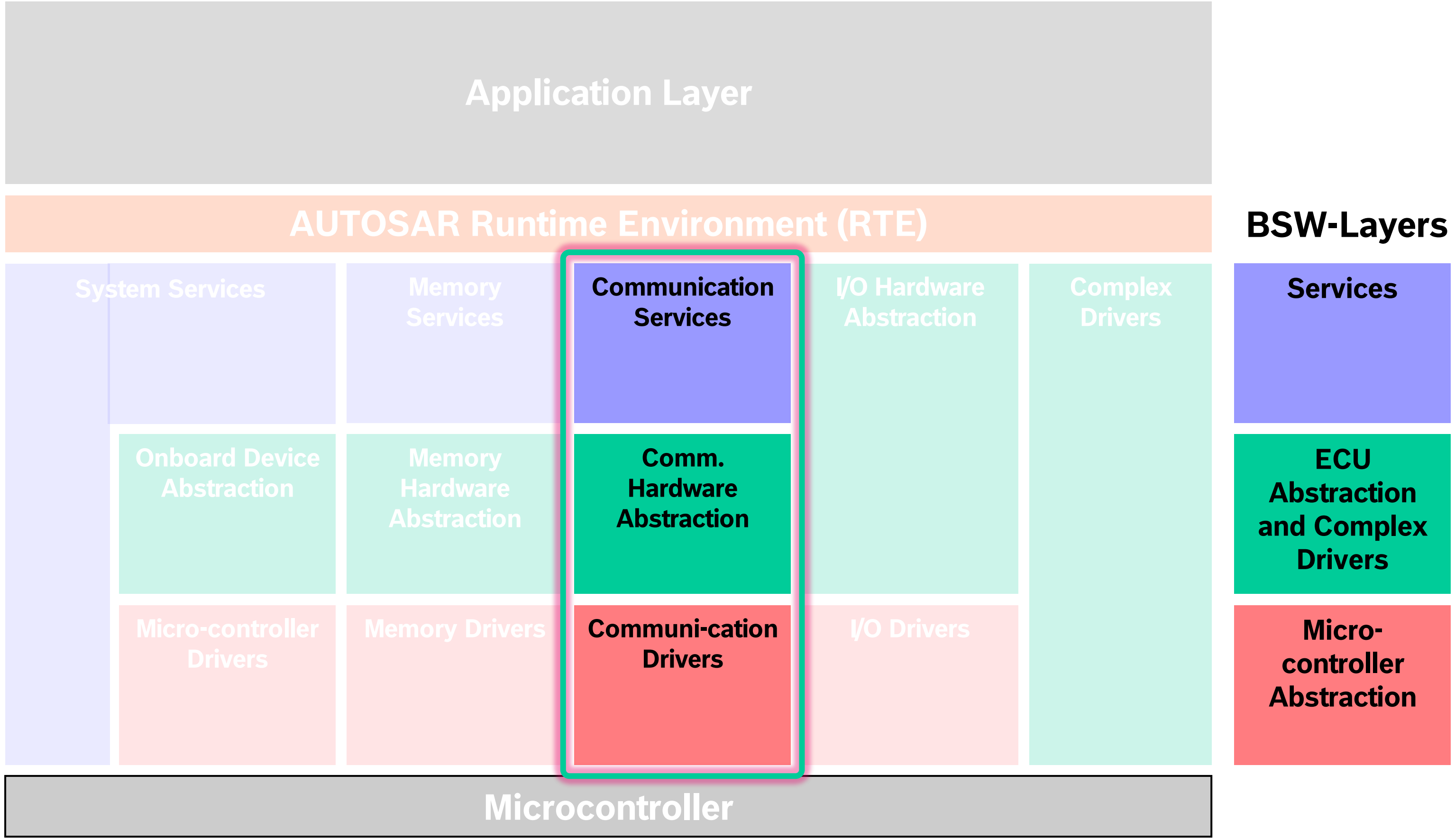


AUTOSAR Basic Software
BSW Vertical view / Stack view



AUTOSAR BASIC SOFTWARE: COMMUNICATION STACK

AUTOSAR Basic Software
Communication Stack – Building Blocks



ComStack facilitates vehicle network communication and provides communication services to other BSW components and Application Layer.

ComStack is consisting of Bus-dependent and Bus-independent components. The following bus system are covered by AUTOSAR:
- CAN(TT)
- LIN
- FlexRay
- Ethernet

Different bus system follows the same layered architecture but defines their components distinctly. *CAN is chosen to represent ComStack in this document.*

AUTOSAR Basic Software

Communication Stack – Building Blocks (cont.)

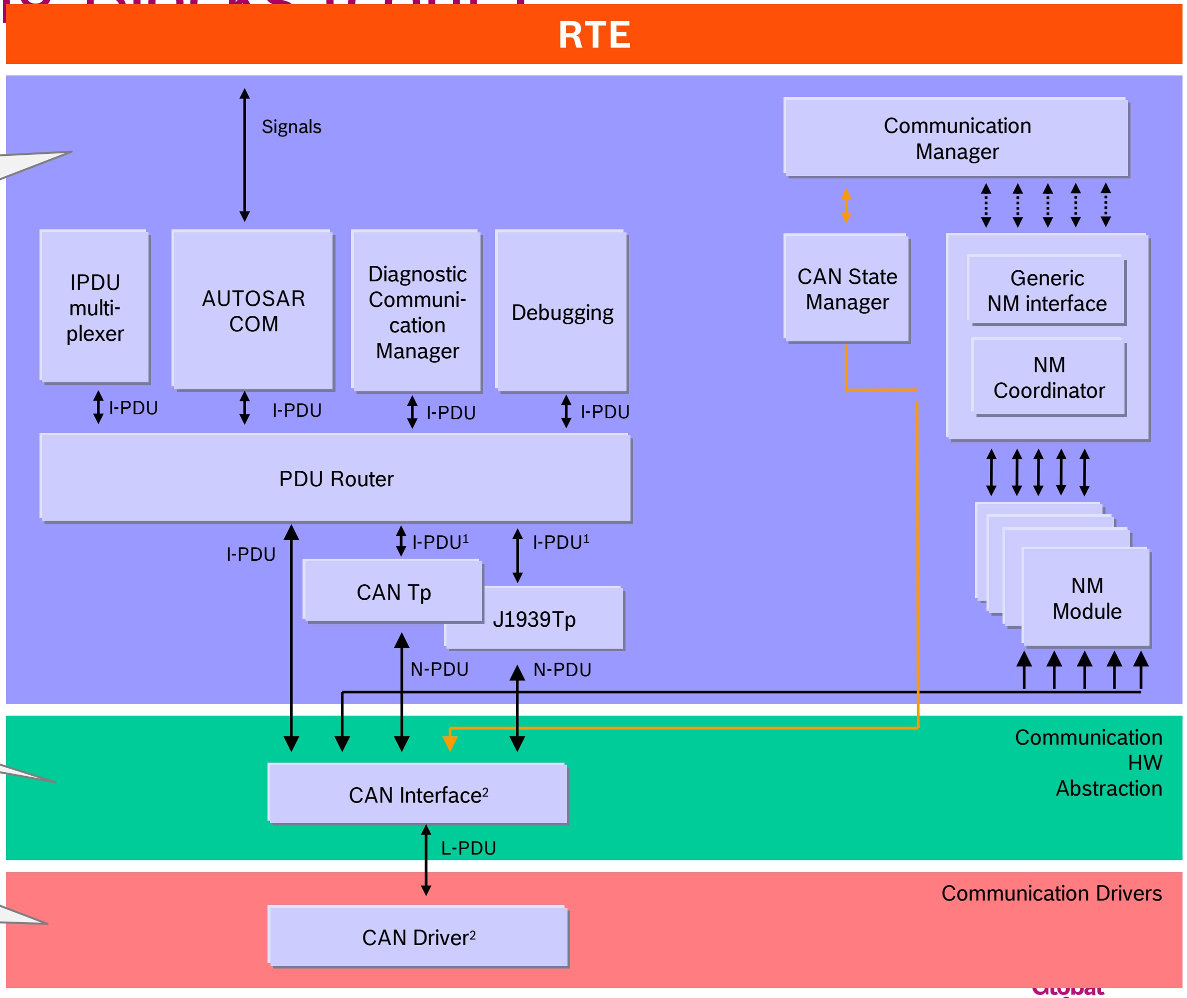
REF

Layers

Layer Communication Services is a group of modules for vehicle network communication with the communication system CAN.
Task:
- Provide a uniform interface to the CAN network
- Hide protocol and message properties from the application.

Layer Communication Hardware Abstraction is a group of modules which abstracts from the location of communication controllers and the ECU hardware layout.
- Provide equal mechanisms to access CAN bus channel regardless of it's location (on-chip / on-board)
- µC independent, ECU hardware dependent and external device dependent

Layer Microcontroller Abstraction contains internal drivers, which are software modules with direct access to the µC internal peripherals and memory mapped µC external devices.
- Make higher software layers independent of µC



AUTOSAR Basic Software

Communication Stack – Building Blocks (cont.)

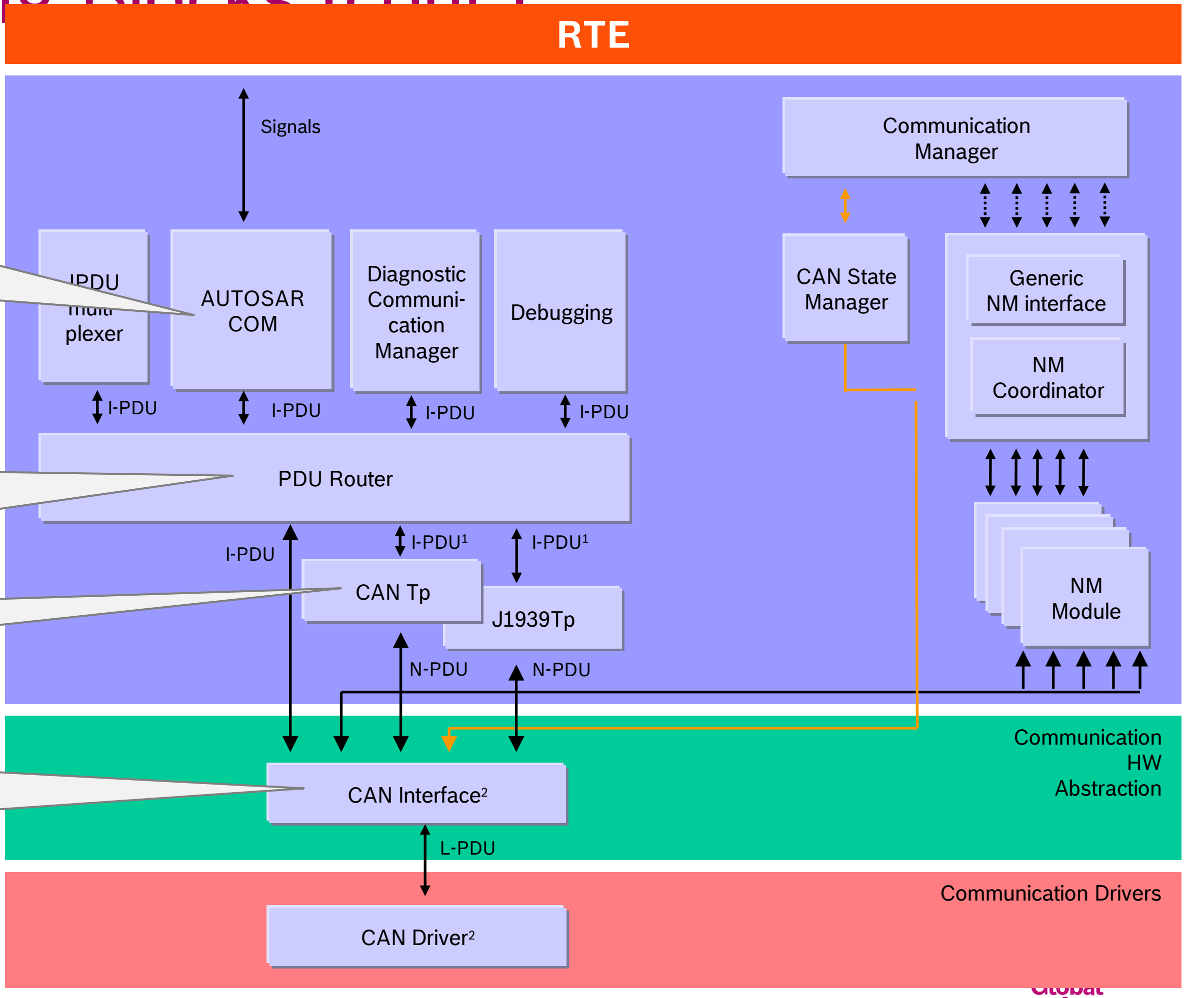
Components

Com
- Provides signal-oriented data interface to the RTE
- Packing/unpacking of AUTOSAR signals to I-PDUs
- Provides routing of individual signals or groups of signals between different I-PDUs

PduR
- Provides routing of PDUs between different abstract communication controllers and upper layers
- Provides TP routing on-the-fly. Transfer of TP data is started before full TP data is buffered

<BusType>Tp
- The main purpose of the TP module is to segment and reassemble (CAN) I-PDUs longer than 8 bytes

<BusType>If
- Provides a unique interface to manage different hardware device types e.g., CAN controllers and CAN transceivers used by the defined ECU hardware layout



REF

AUTOSAR Basic Software

Communication Stack – Building Blocks (cont.)

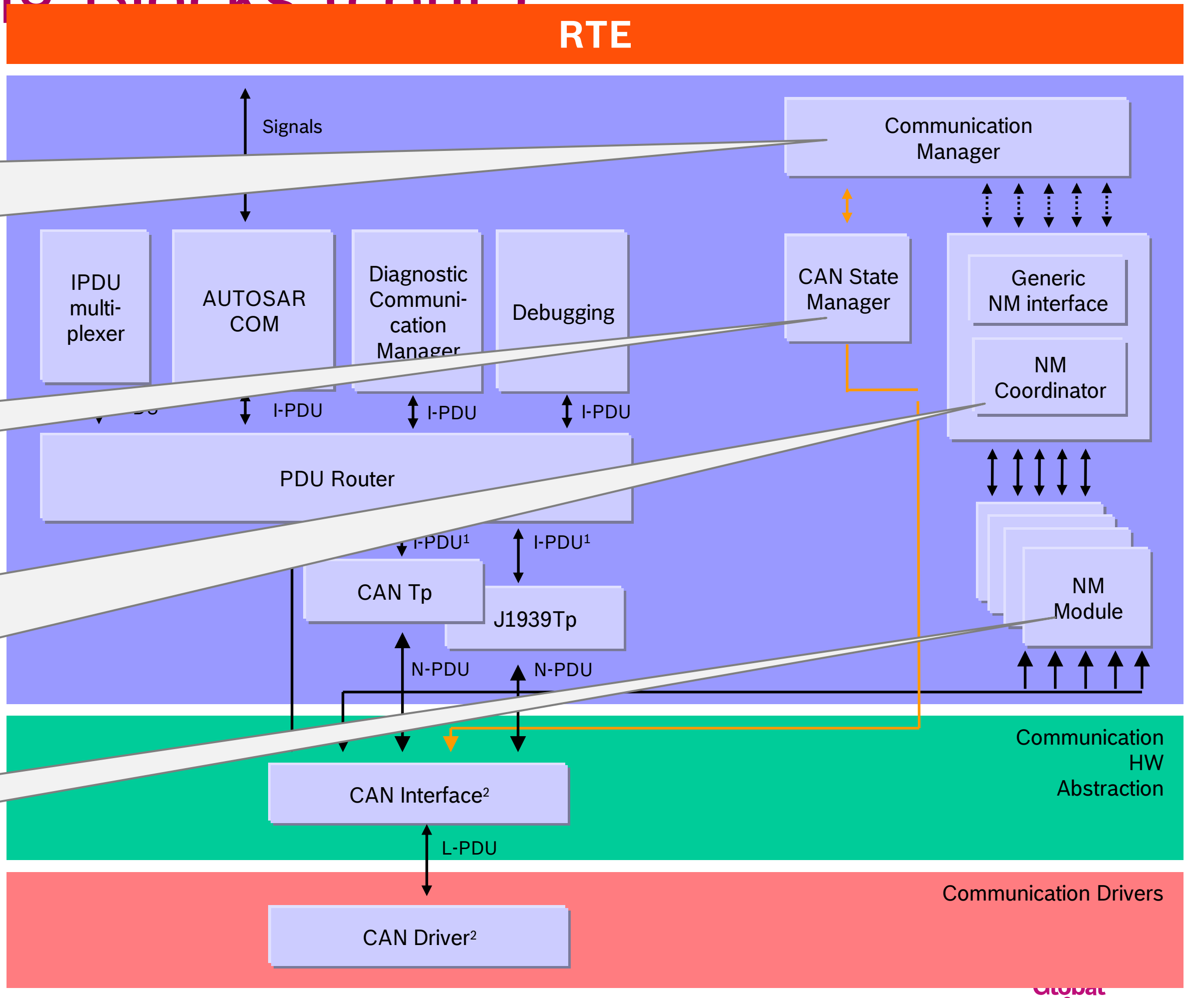
Components

ComM
- Collects the bus communication access requests from communication requestors (Applications, Complex Drivers) and coordinates the bus communication access requests.
- Triggers the Start-up and Shut-down the hardware units of the communication systems

<BusType>SM
- Handles the communication system dependent Start-up and Shutdown features

Nm
- Acts as a bus-independent adaptation layer between the bus-specific Network Management modules and the Communication Manager module (ComM)
- Synchronization of Network States of different communication channels connected to an ECU via the network managements handled by the NM Coordinator

<BusType>Nm
- Coordinate the transition between normal operation and bus-sleep mode of the network.



Global Software Technologies alt_future

REF

AUTOSAR Basic Software

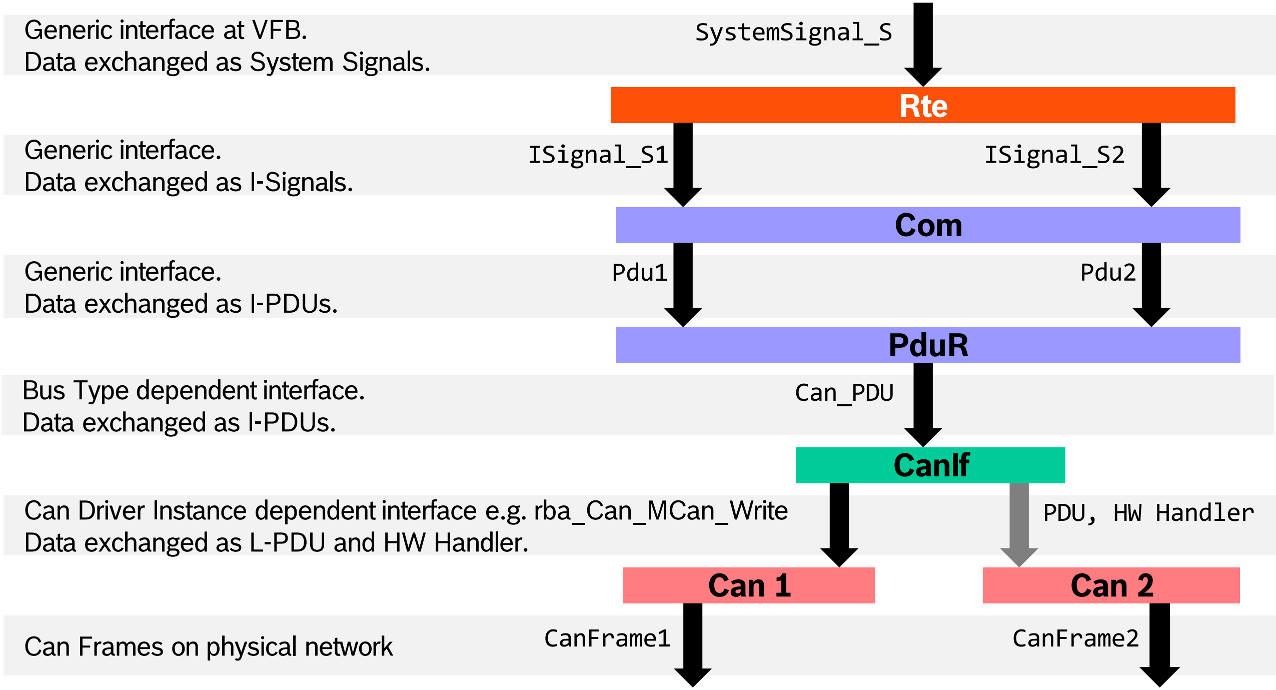
Communication Stack – Runtime: transmission and reception

System Signals and ISignals are introduced to distinguish between the unique pieces of data transferred between SWCs and the interaction layer signal used to distribute this data to multiple receivers.

A PDU (Protocol Data Unit) is the information delivered through a network layer

A Frame is a piece of information that is exchanged over the communication channels

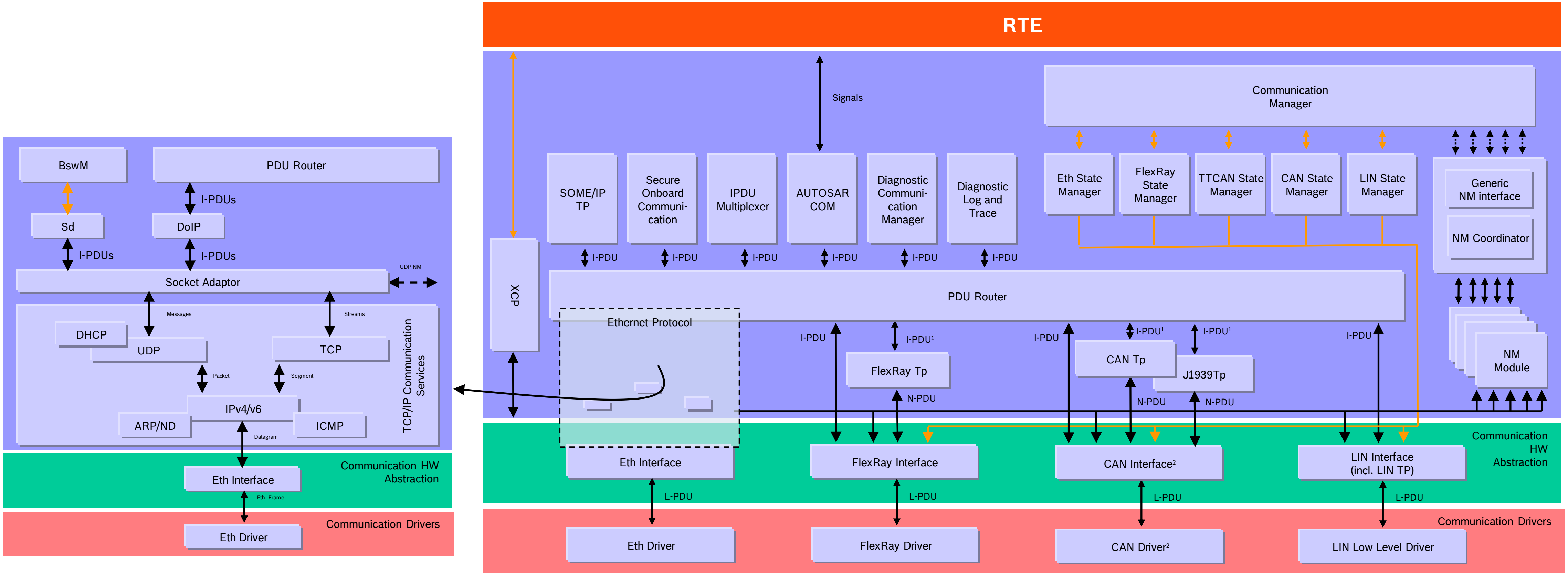
ISO Layer	Layer Prefix	AUTOSAR Modules	PDU Name
Layer 6: Presentation (Interaction)	I	COM, DCM	I-PDU
	I	PDU router, PDU multiplexer	I-PDU
Layer 3: Network Layer	N	TP Layer	N-PDU
Layer 2: Data Link Layer	L	Driver, Interface	L-PDU



Bosch Global Software Technologies
alt_future

AUTOSAR Basic Software
Communication Stack – Runtime (cont.)

REF



I-PDU: Interaction Layer PDU
N-PDU: Network Layer PDU
L-PDU: Data Link Layer PDU

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.

AUTOSAR Basic Software
Communication Stack – OSEK COM Layer Model

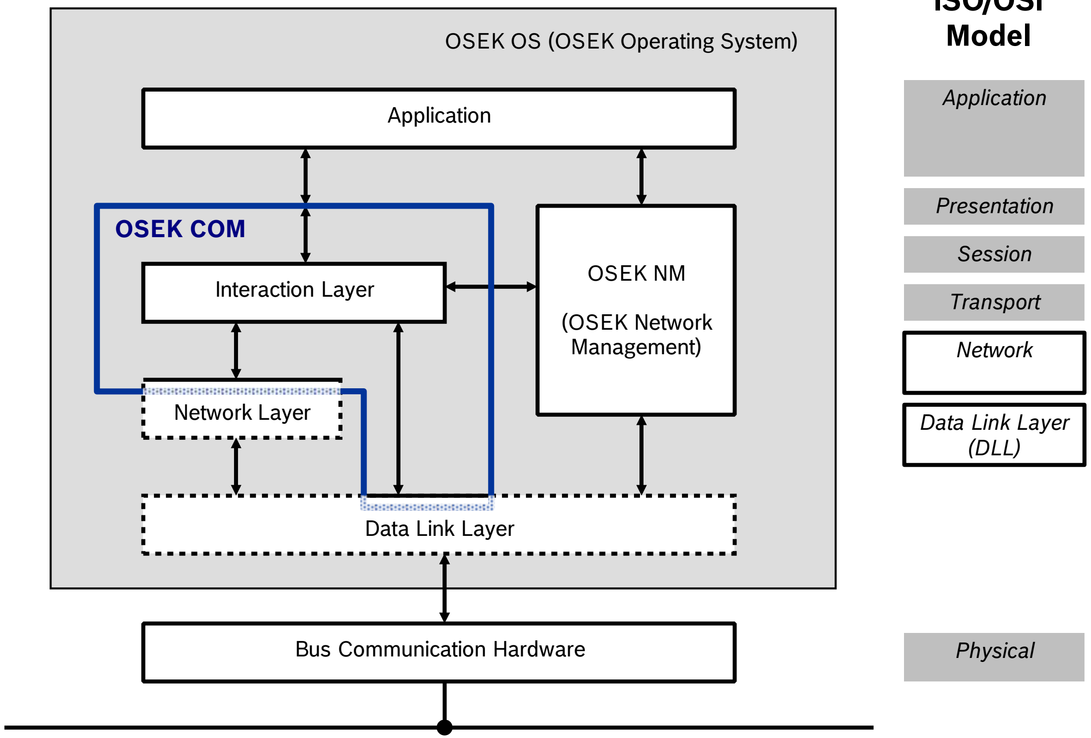
REF

It's the foundation of AUTOSAR ComStack!

Interaction Layer (IL)
- Provides the OSEK COM API which contains services for the transfer (send and receive operations) of messages.
- For external communication it uses services provided by the lower layers, whereas internal communication is handled entirely by the IL.

Network Layer
- Handles – depending on the communication protocol used – message segmentation/recombination and acknowledgement.
- Provides flow control mechanisms to enable the interfacing of communication peers featuring different levels of performance and capabilities.
- The Network Layer uses services provided by the Data Link Layer.
- OSEK COM does not specify the Network Layer; it merely defines minimum requirements for the Network Layer to support all features of the IL.

Data Link Layer
- Provides the upper layers with services for the unacknowledged transfer of individual data packets (frames) over a network.
- Provides services for the NM.
- OSEK COM does not specify the Data Link Layer; it merely defines minimum requirements for the Data Link Layer to support all features of the IL.



OSEK: Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen (DE) or Open Systems and the Corresponding Interfaces for Automotive Electronics (EN).

AUTOSAR Layered Architecture
Quiz time

- **Which component is responsible for distribution of Pdu for inter- and intra-ECU communication?**

(a) Com.

(b) PduR.

(c) ComM.

(c) Rte.

AUTOSAR Layered Architecture
Quiz time

- **Which component is responsible for distribution of Pdu for inter- and intra-ECU communication?**

(a) Com.

(b) PduR.

(c) ComM.

(c) Rte.

AUTOSAR Layered Architecture
Quiz time

- **Which statement below is not correct when talking about data exchange in AUTOSAR?**

(a) On Application layer, System Signal is being exchanged without context of bus system.

(b) Two Signal can be packed inside a Pdu. Two Pdus can be packed inside a Frame.

(c) I-Pdu is data unit at Interaction layer. L-Pdu is data unit of Data-link layer.

(c) N-Pdu is data unit at Transport layer, which is mandatory for all AUTOSAR systems.

AUTOSAR Layered Architecture
Quiz time

- **Which statement below is not correct when talking about data exchange in AUTOSAR?**

(a) On Application layer, System Signal is being exchanged without context of bus system.

(b) Two Signal can be packed inside a Pdu. Two Pdus can be packed inside a Frame.

(c) I-Pdu is data unit at Interaction layer. L-Pdu is data unit of Data-link layer.

(c) N-Pdu is data unit at Transport layer, which is mandatory for all AUTOSAR systems.

Bosch Global Software Technologies
alt_future

AUTOSAR Layered Architecture
Quiz time

- **What are the drawbacks of AUTOSAR?**

(a) Standardization by aggregation.

(b) It is huge.

(c) Moving target.

(c) All of above.

AUTOSAR Layered Architecture
Quiz time

- **What are the drawbacks of AUTOSAR?**

(a) Standardization by aggregation.

(b) It is huge.

(c) Moving target.

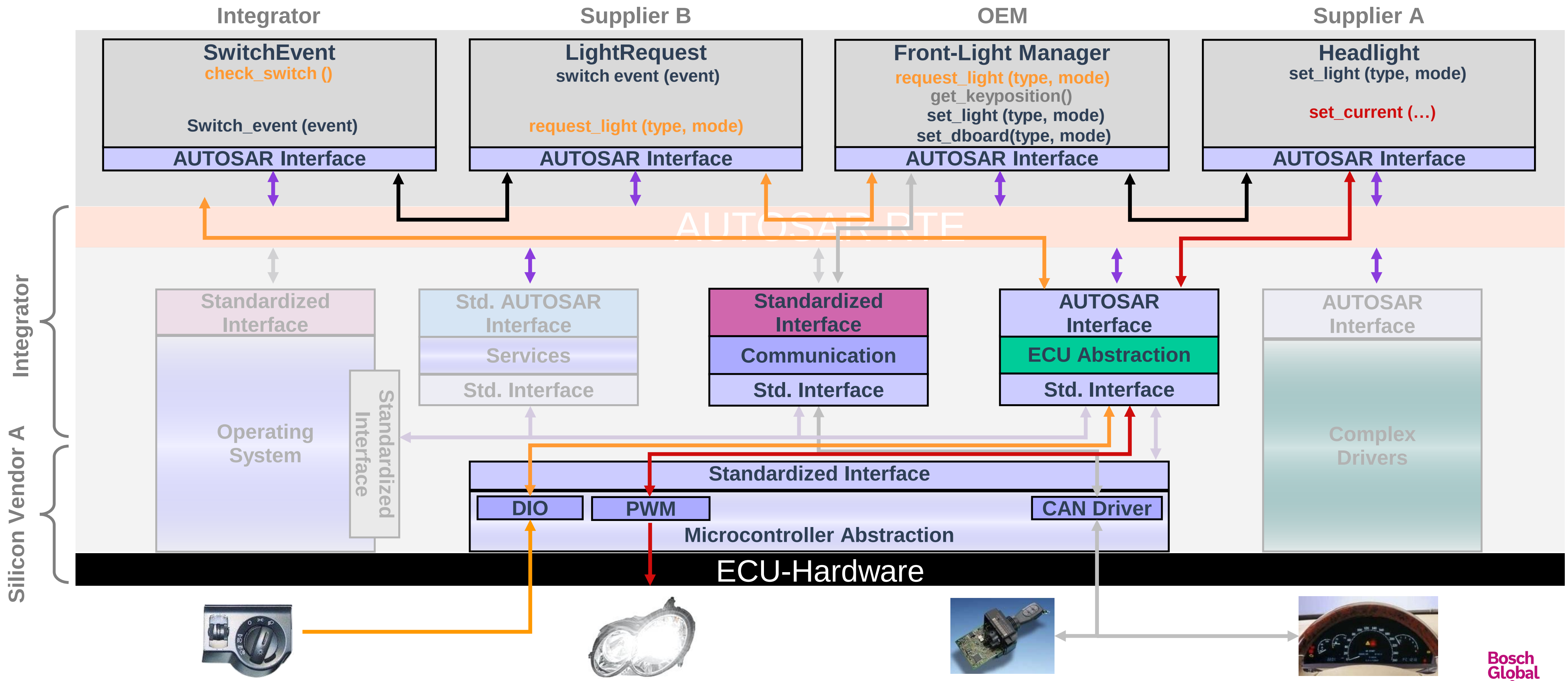
(c) All of above.

CLASSIC AUTOSAR USE CASE

Source: AUTOSAR Introduction, AUTOSAR Consortium

Classic AUTOSAR Use Case

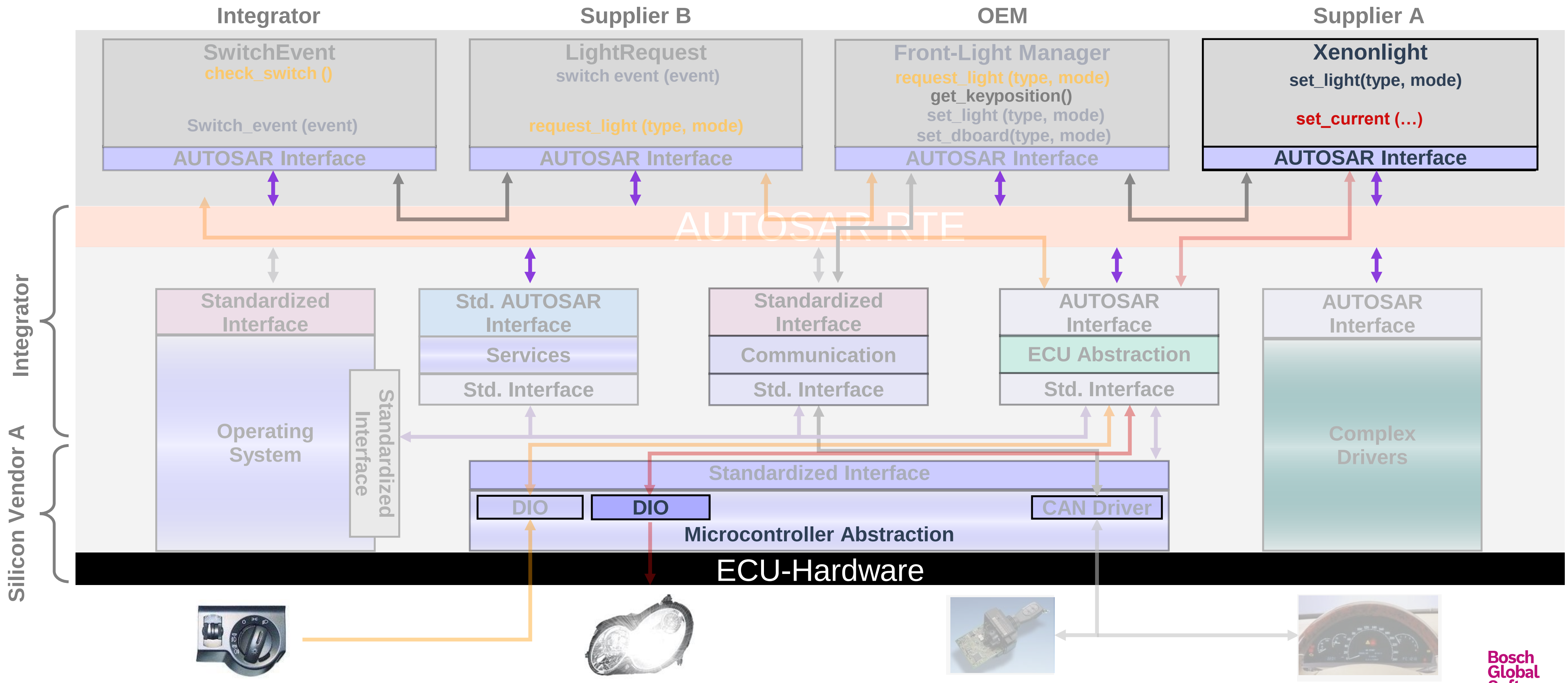
Use Case ‘Front Light Management’: Exchange Type of Front Light



Bosch Global Software Technologies
alt_future

Classic AUTOSAR Use Case

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



Bosch Global Software Technologies alt_future

Classic AUTOSAR Use Case
Distribution ECUs

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)

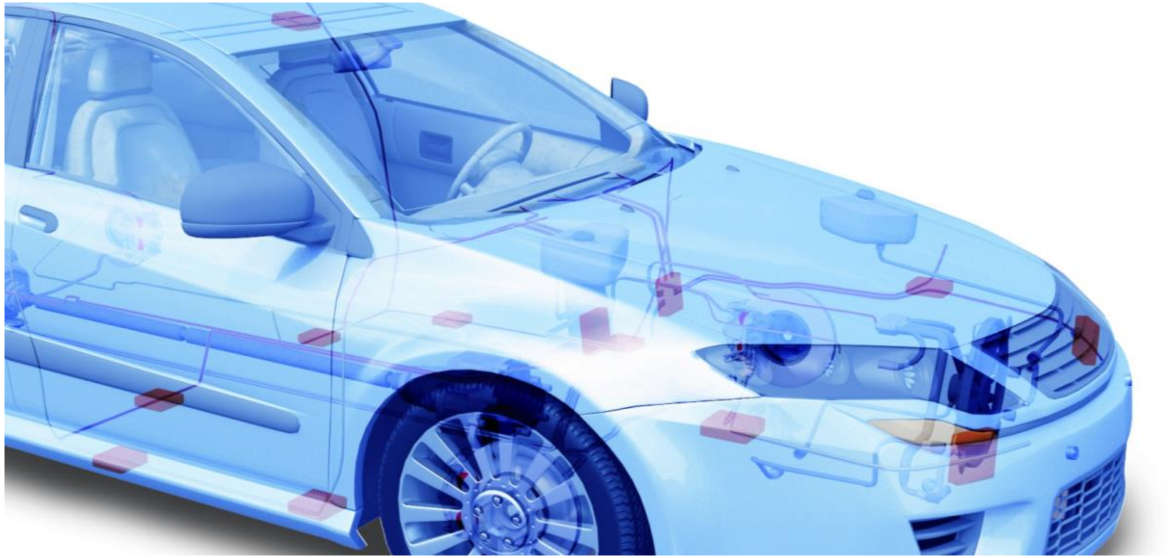
AUTOSAR Interface

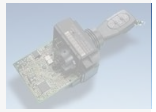
Xenonlight

set_light(type, mode)

set_current (…)

AUTOSAR Interface

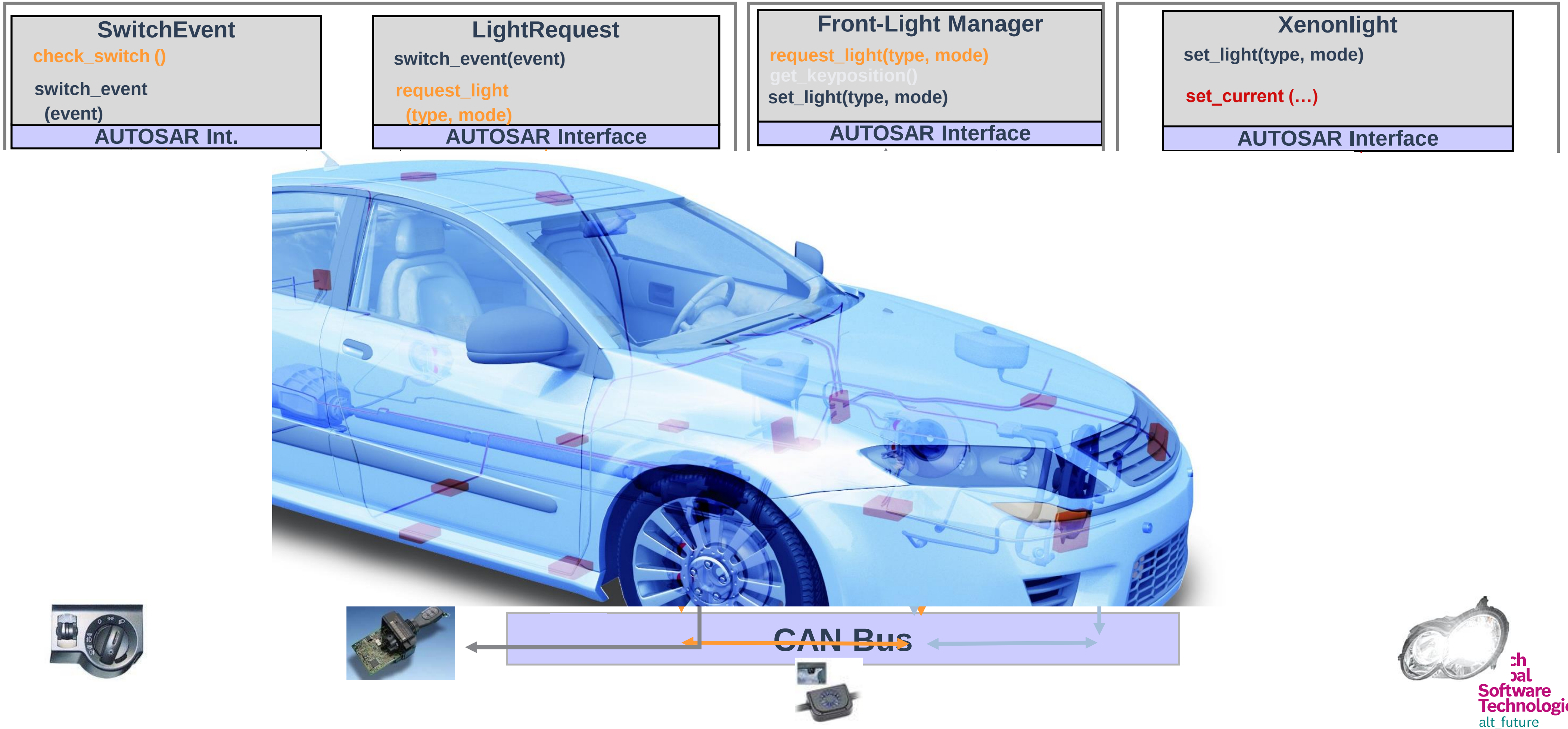




Bosch Global Software Technologies
alt_future

Classic AUTOSAR Use Case

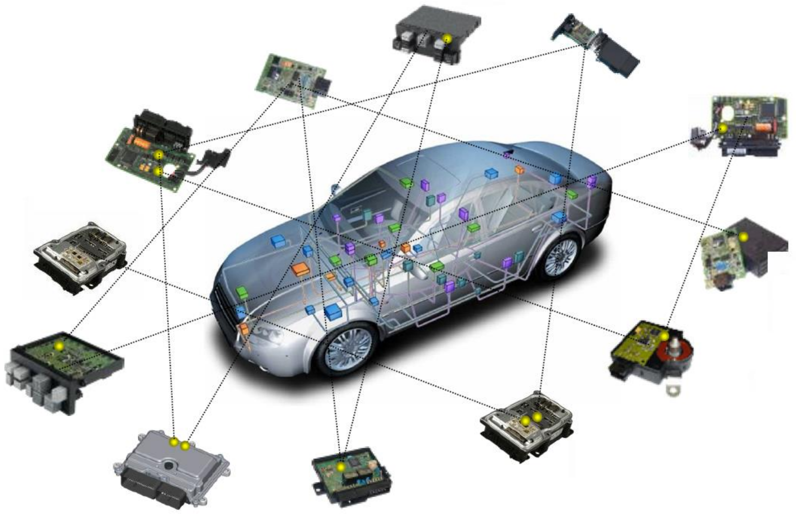
Distribution on ECUs – ‘Front-Light Management’

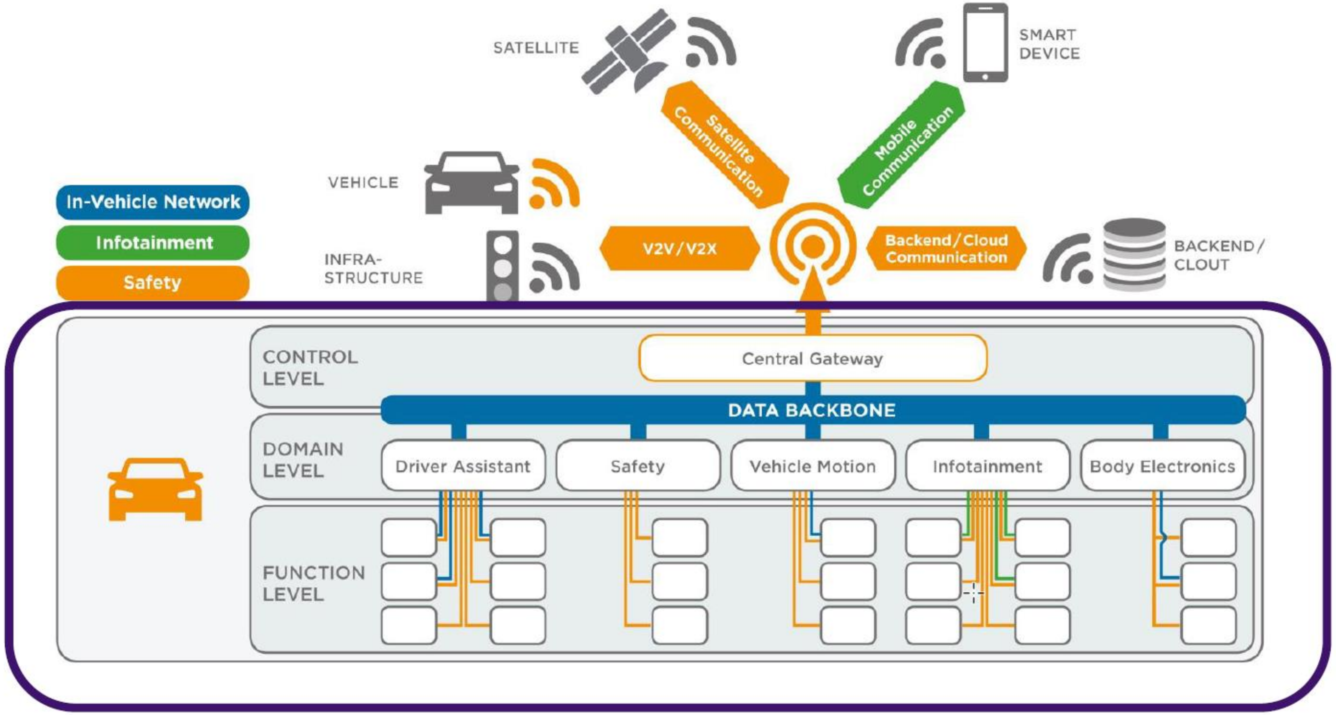


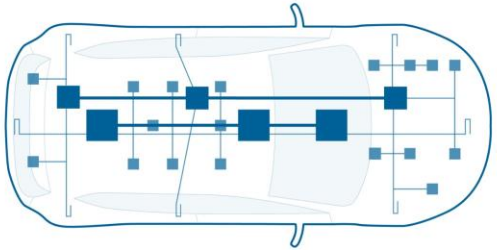
Software Technologies
alt_future

CAN
PROTOCOL

Introduction
Vehicle network

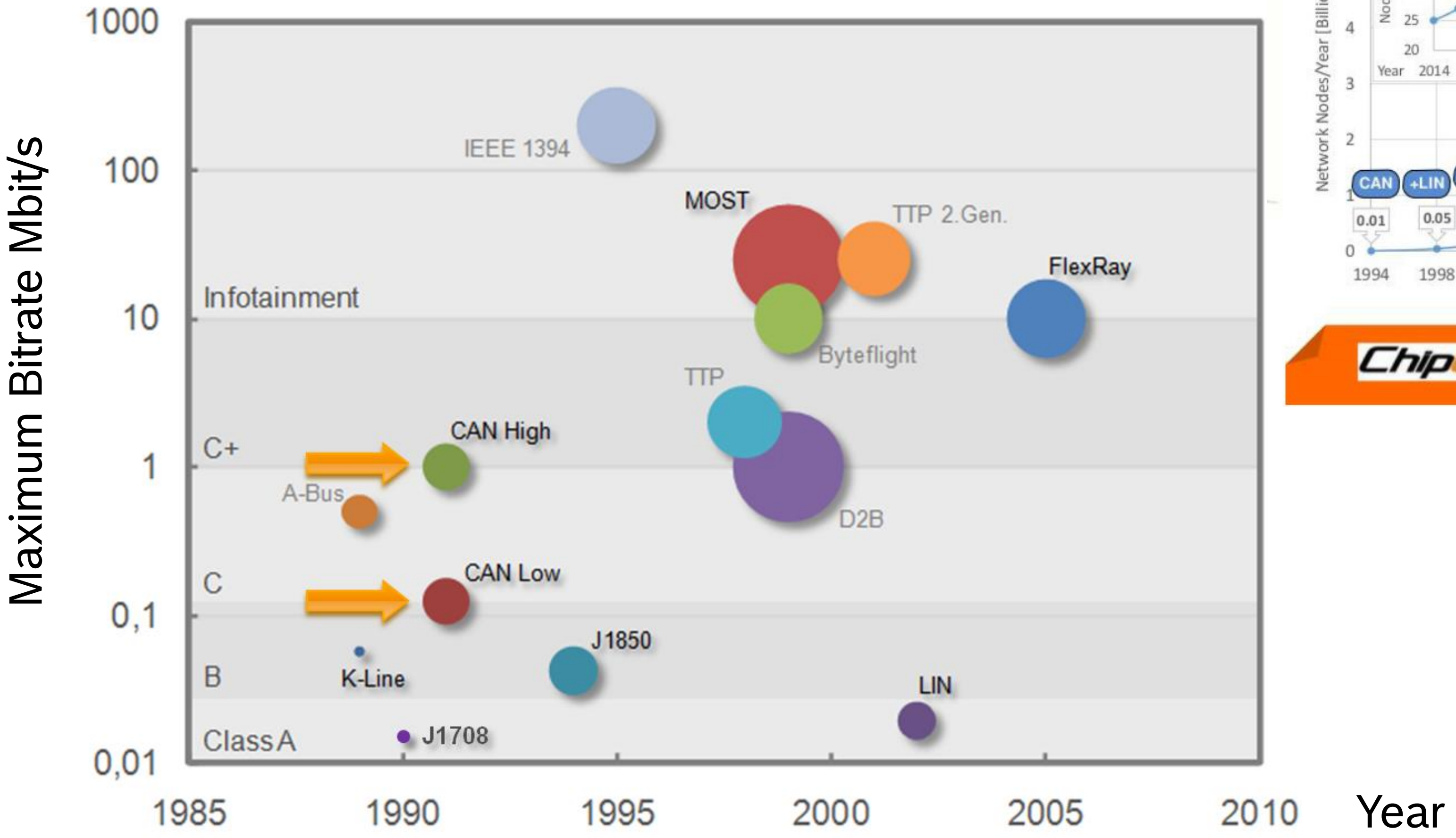


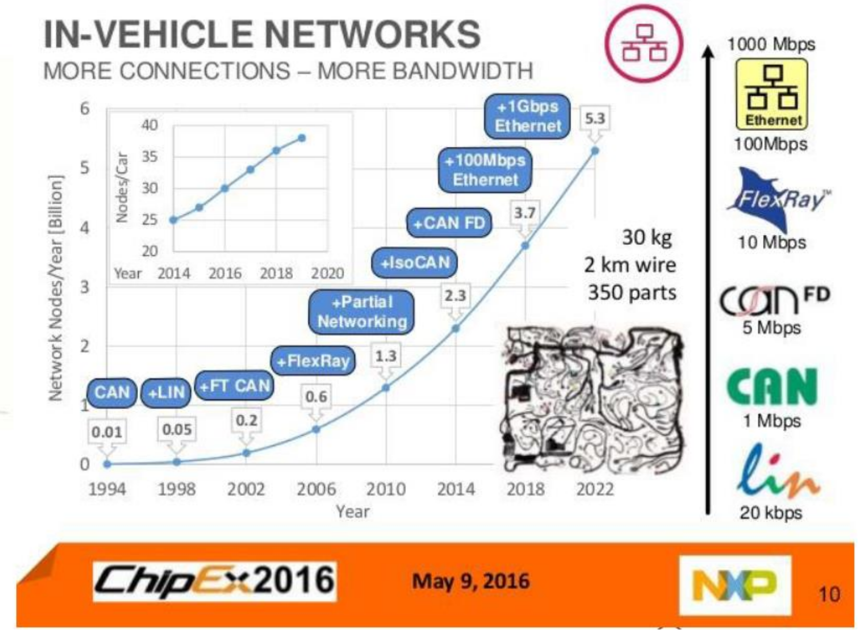




Bosch Global Software Technologies
alt_future

Introduction
History



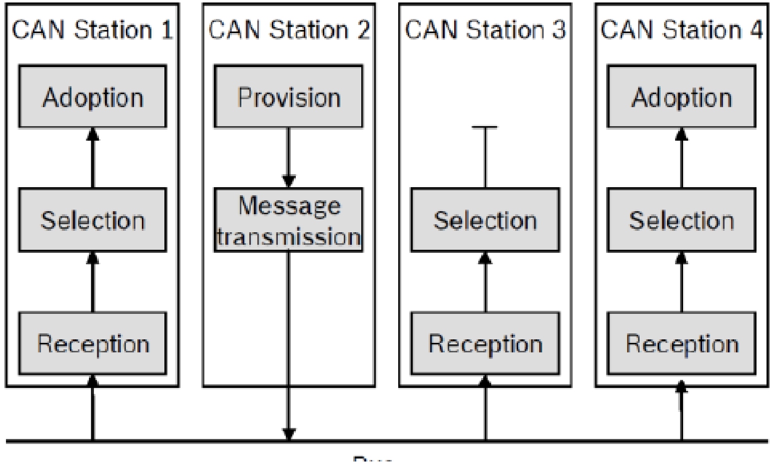


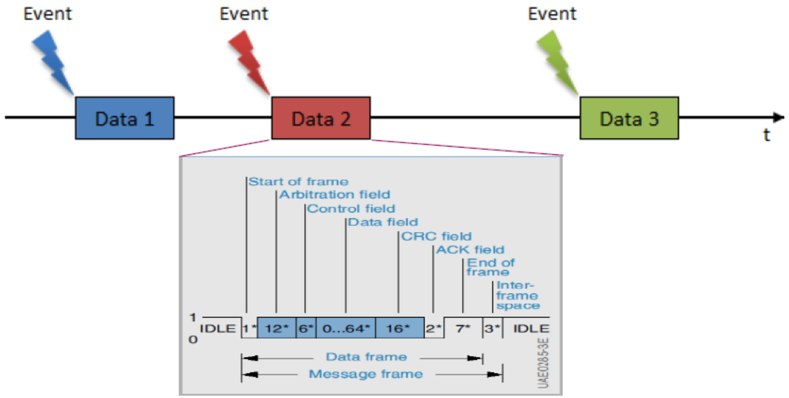
Bosch Global Software Technologies
alt_future

Introduction

Characteristics of ‘CAN’

- CAN is a multi-master Bus
- Theoretically No limitation on the number of nodes
- Configuration flexibility - No node addressing
- Prioritization of messages through “Identifiers”
- Multicast reception with the time synchronization
- System wide data consistency
- Guarantee of latency times
- Error detection and error signaling
- Automatic retransmission of corrupted messages
- Temporary errors - permanent failures of nodes and a
switching off defect nodes





Bosch Global Software Technologies
alt_future

Introduction
CAN in the OSI model

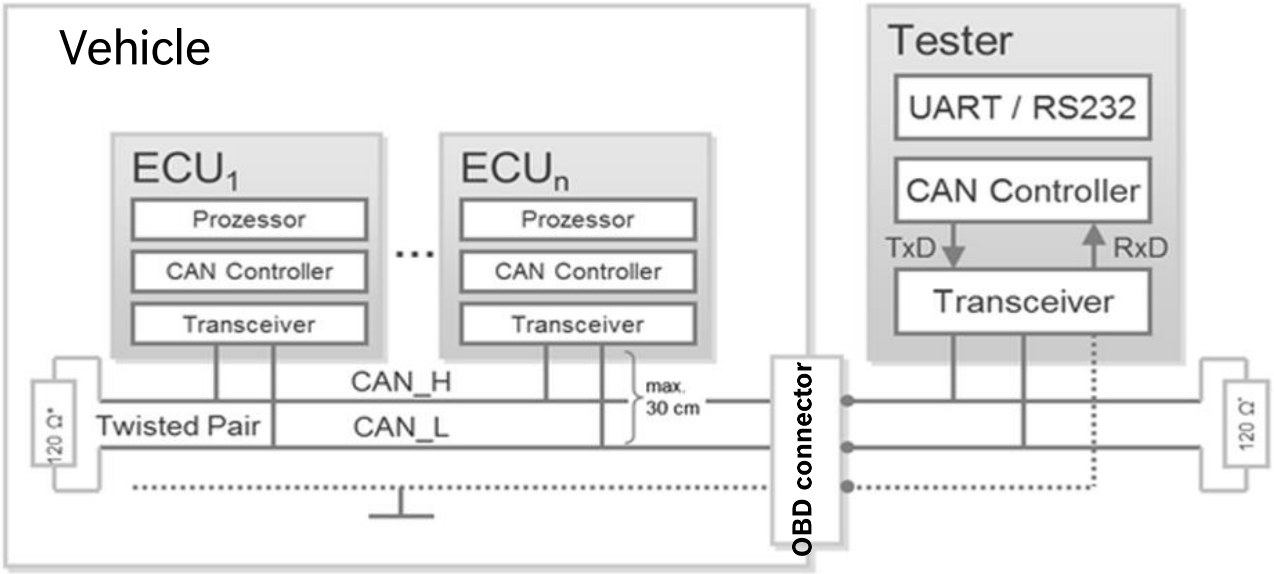
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

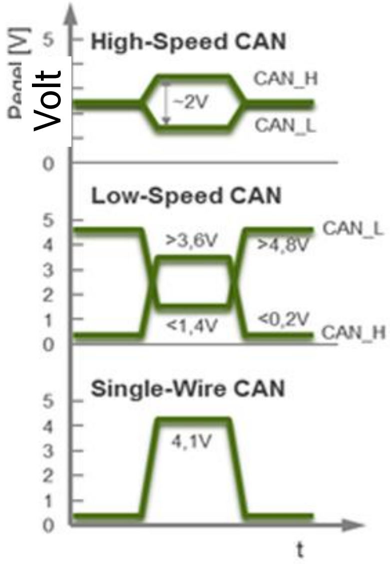
- Logical Link Control (LLC)
  - Acceptance Filtering
  - Overload Notification
  - Recovery Management
- Medium Access control(MAC)
  - Data Encapsulation/Decapsulation
  - Frame Coding
  - Error Detection/Signaling/Handling

- Physical Signaling (PLS)
  - Bit Encoding/Decoding
  - Bit Time Synchronization
- Physical Medium attachment(PMA)
  - Driver/Receiver Characteristics
- Media Dependant Interface(MDI)
  - Connectors

CAN Protocol
Physical Layer

- Bit rate: up to 1Mbit/s
- Bidirectional Dual-wire bus with 40-50m maximum in length
- Multi-Master



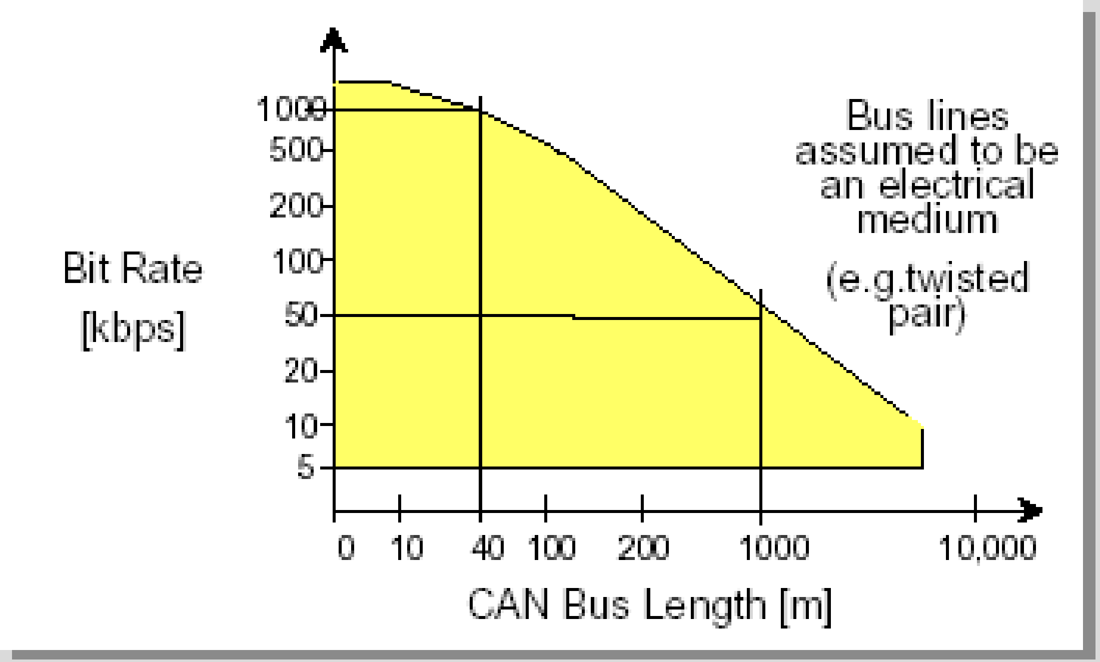


Bosch Global Software Technologies
alt_future

CAN Protocol

Relation between Baud Rate and Bus Length

- Up to 1Mbit / sec @40m bus length (130 feet)

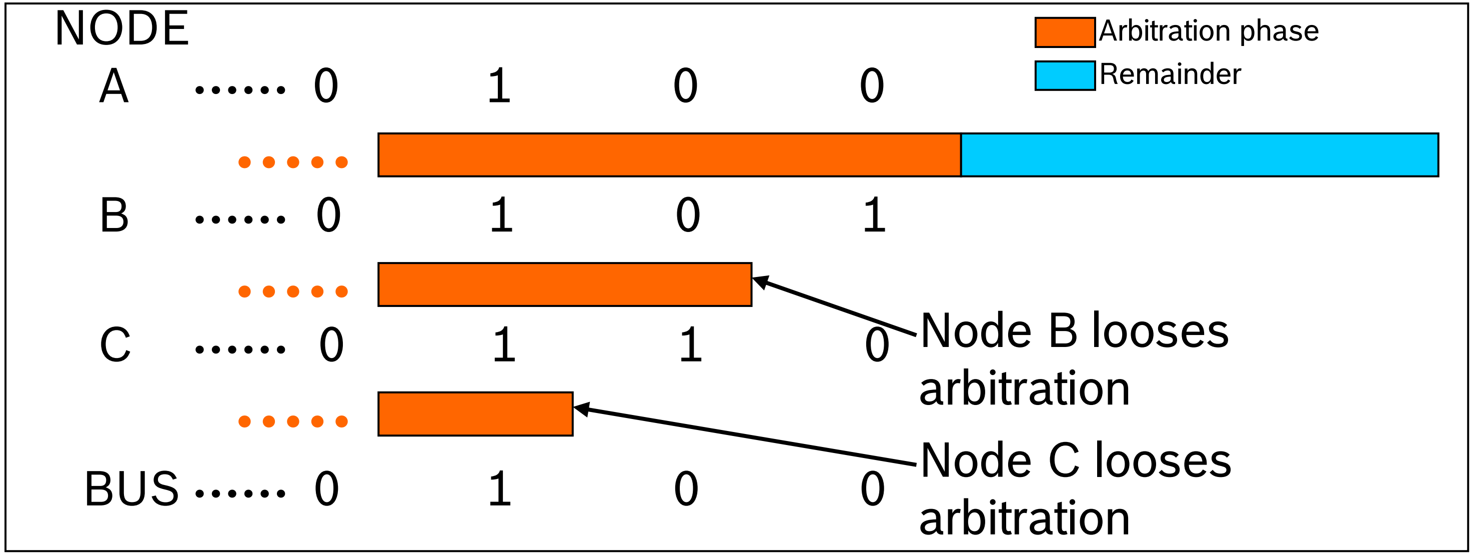


Bosch Global Software Technologies
alt_future

CAN Protocol

Bus Access and Arbitration

- Bus access through CSMA with AMP



- Advantages
  - No Collision
  - Transmission of highest priority message within the latency time

Bosch Global Software Technologies alt_future

CAN Protocol
Message Transfer

Frame Formats

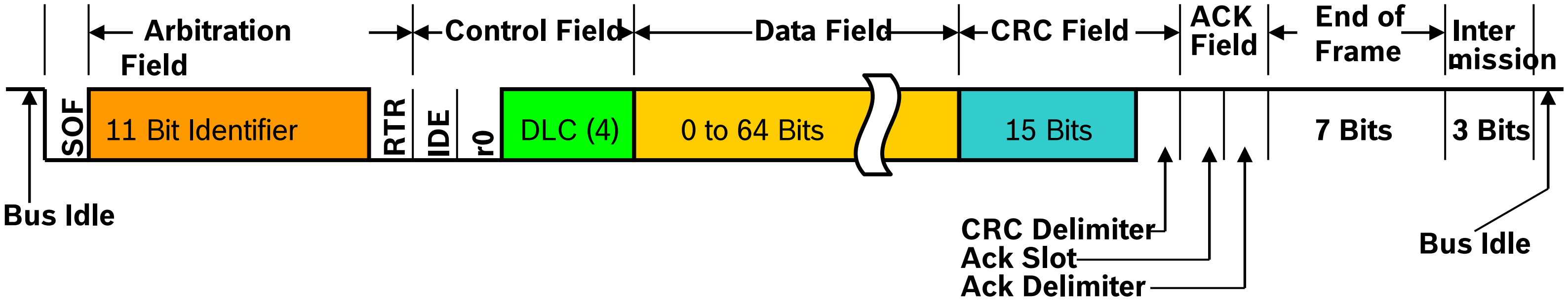
- Standard Frame - 11bit Identifier
- Extended Frame - 29 bit Identifier

Frame Types

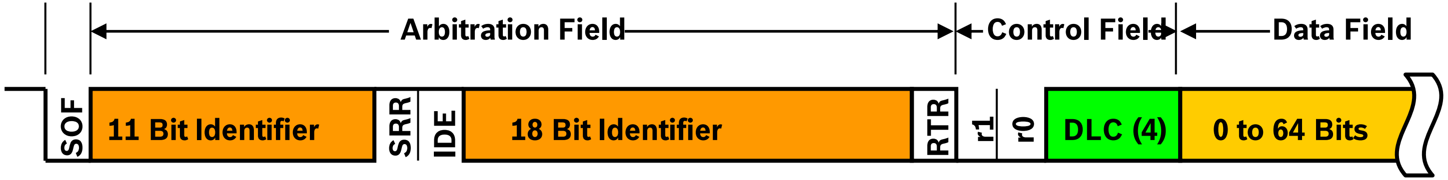
- Data Frame
- Remote Frame (not useful)
- Error Frame
- Overload Frame (not useful)
- Inter-frame Spacing

CAN Protocol
Data Frame

Standard Data Frame Format



Extended Data Frame Format

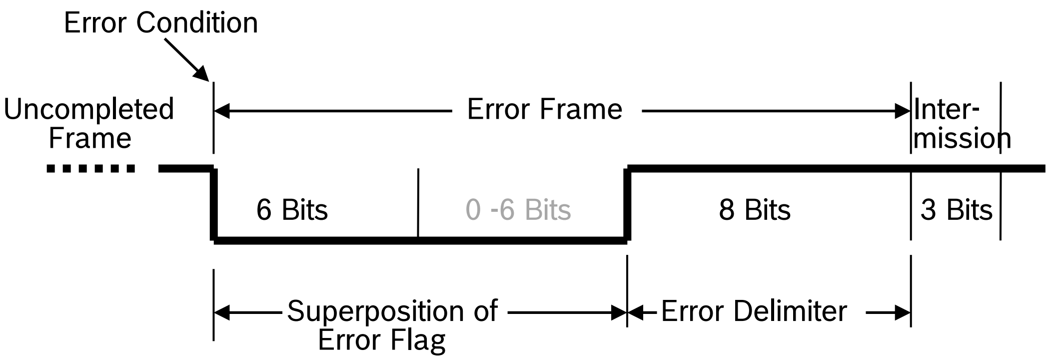


Difference between Standard Frame and Extended Frame

- **Differs only in Arbitration field and Control field**

CAN Protocol
Error Frame

Error Frame Format (Active Error Frame)

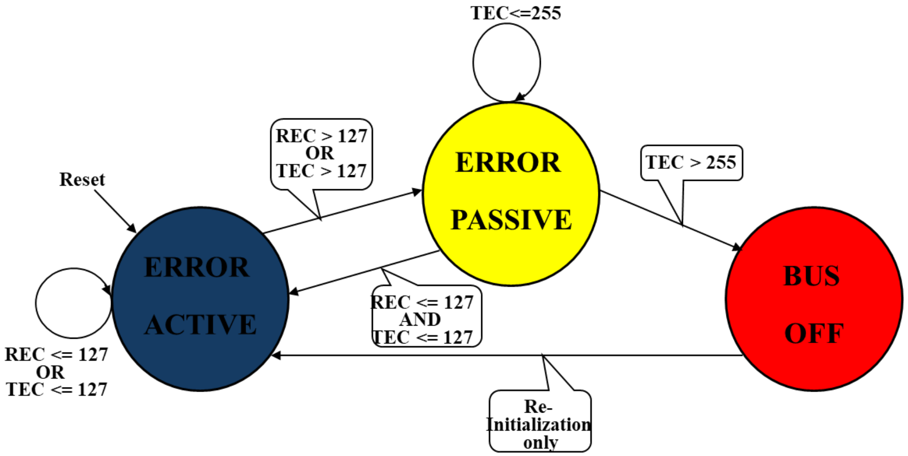


- Error flag can start within the frame that is currently being transmitted

Types of Error flags

- Active Error flag - consists of 6 consecutive ‘dominant’ bit
- Passive Error flag - consists of 6 consecutive ‘recessive’ bit

CAN Protocol
Error Handling



The mode of the controller is controlled by two error counters - the transmit error counter (tx_count) and the receive error counter (rx_count).

The following rules apply:

- The CAN controller is in **error active mode** if:

tx_count <= 127 AND rx_count <= 127.

- Passive mode is used if :

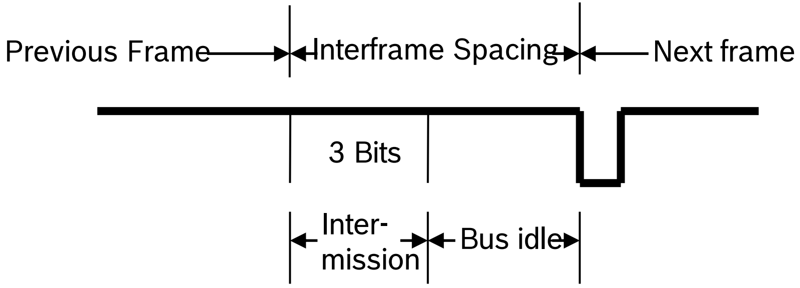
tx_count > 127 or rx_count>127 AND tx_count <= 255.

- Bus off is entered if:

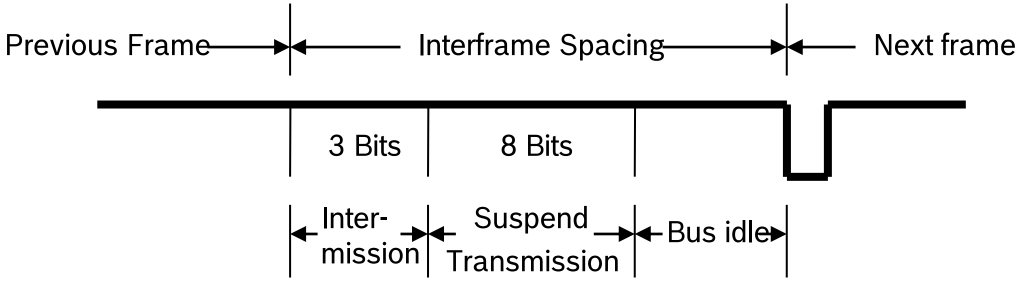
tx_count > 255.

CAN Protocol
Interframe Spacing

After the transmission of a frame by an Error Active node



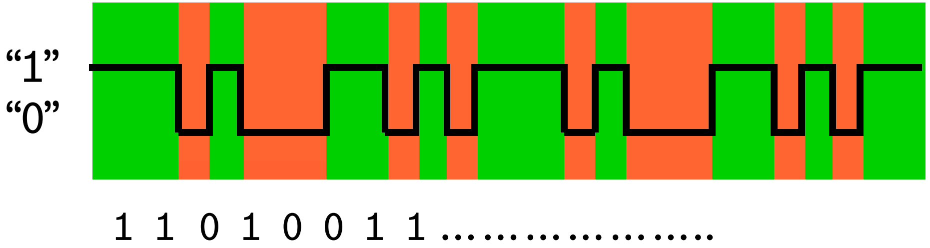
After the transmission of a frame by an Error Passive node



CAN Protocol
Message Coding

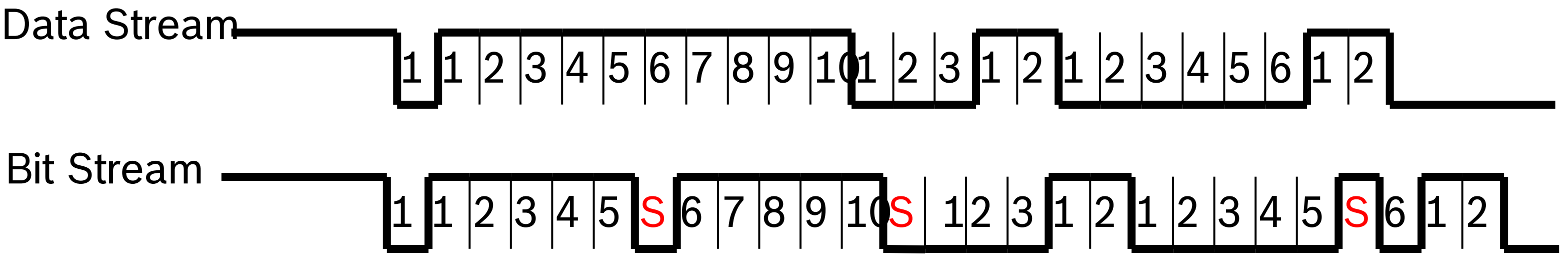
Non-Return-to-Zero coding

- Keeps the frequency of the signal on the bus to minimum.



Bit-Stuffing

- Ensures sufficient Recessive and Dominant edges for Re-Synchronization.



CAN Protocol
Types of Error Detected in CAN Bus

CRC Error:

- Every node receive the message, Calculate CRC and compare it with Received CRC.

Acknowledge Error:

- Transmitting node send a ACK slot bit as a recessive bit and check for dominant bit to verify reception.

Form Error:

- Generated when any of following bit is detected as a dominant bit where One should not be.

e.g. CRC delimiter, ACK delimiter, End of Frame, Inter Frame Space.

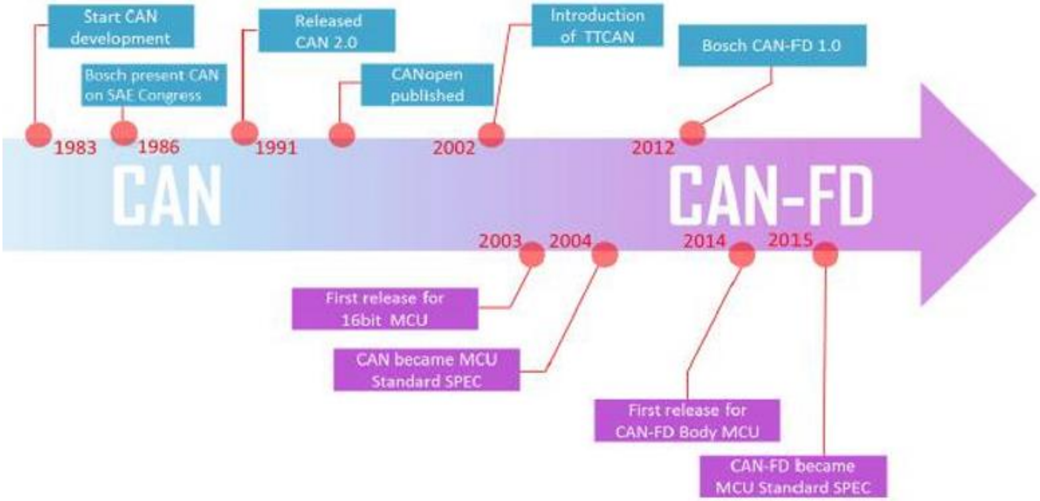
Bit Error:

- Node detect the signal that is opposite of what it send on Bus.

Stuff Error:

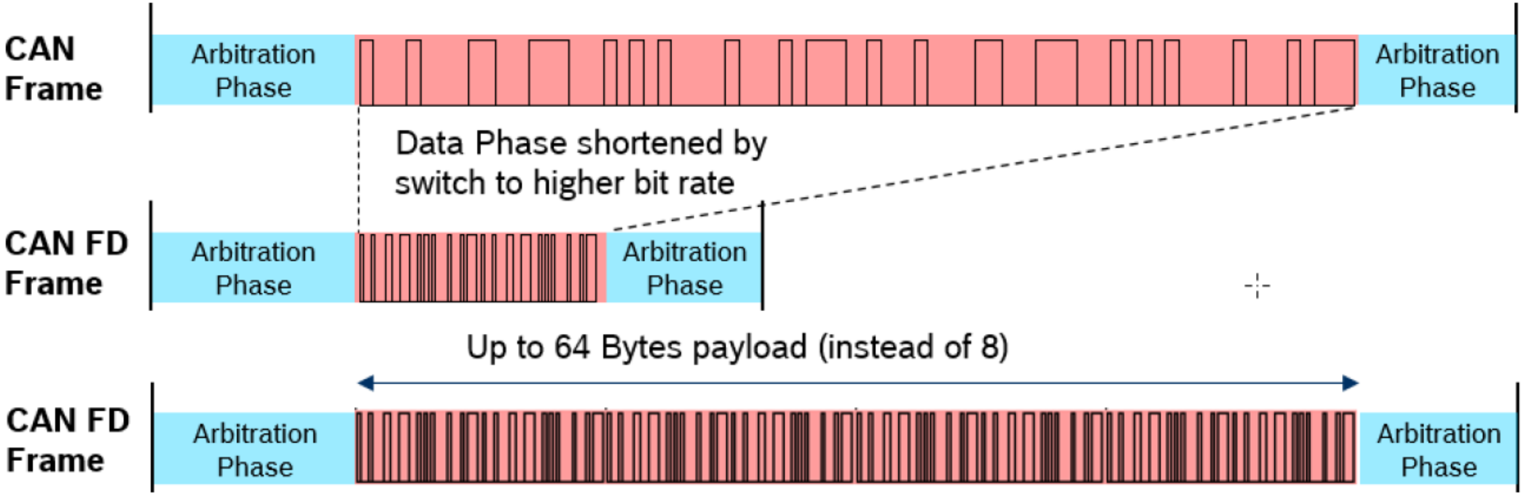
- Bit stuffing rule is violated when 6-consecutive bits with the same polarity are detected.

Introduction about CAN FD



Main improvement:

- Increase bit rate (2,4 ... up to 8 Mbit/s)
- Increase payload up to 64 bytes



Reference

-CAN Specification 2.0 – Bosch

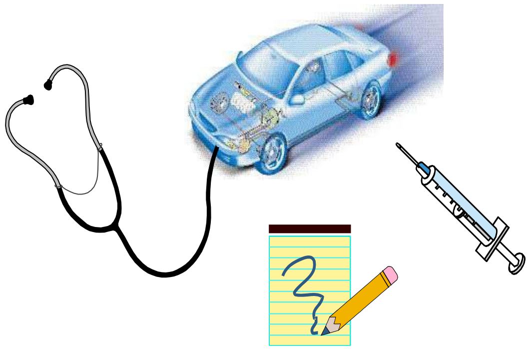
-ISO 11898-2 – High speed CAN

-ISO 11898-2 2015 – CAN FD

Q & A

DIAGNOSIS OVERVIEW

Definitions
Diagnosis – What?



"In automotive engineering, **Diagnosis** is typically used to determine the causes of symptoms and solutions to issues."

- symptom(s) – what the user/operator/repairer of the system (vehicle or whatever) notices;
- fault(s) – the error(s) in the system that result in the symptom(s);
- root cause(s) – the cause(s) of the fault.

Source: Advanced Automotive Fault Diagnosis- Automotive Technology: Vehicle Maintenance and Repair

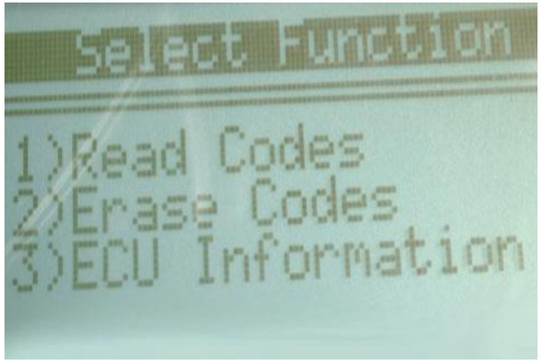
TOM DENTON

Definitions
Diagnosis – How?[1]

To do Diagnostic, Technician have to know how to use Diagnostic Tools and Equipment.

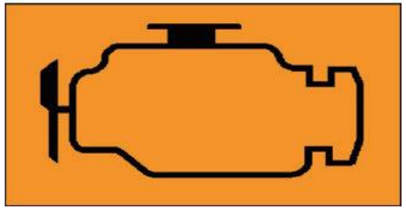
Tool and Equipment could be classified into:

- Basic Equipment: such as Multi-meter
- Tracing Tool: like Oscilloscope
- Scanner/Fault Code Readers and Analyzers.



Definitions
Diagnosis – How?[2]

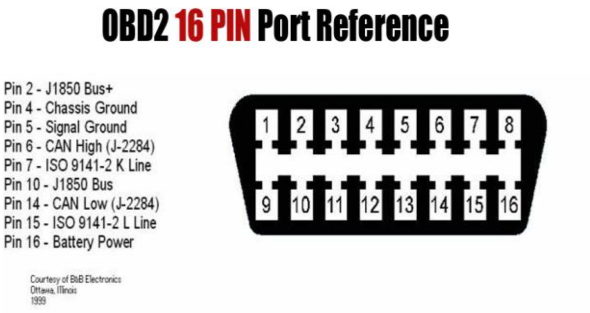
- The Equipment shall help technician indicate where is fault occurs in systems.
- In the other word, In Vehicle, Systems should have ability to provide information in case request.
- This is the motivation of On-board diagnostics (OBD).



- On-board diagnostics (OBD) is a generic term referring to a vehicle’s self-diagnostic and reporting system. OBD systems give the vehicle owner or a technician access to information for various vehicle systems.
- OBD system illuminates a warning lamp known as the malfunction indicator lamp (MIL) or malfunction indicator (MI) on the instrument cluster.

Definitions

Diagnosis – How?[3]



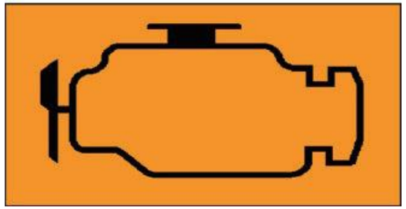
- When the fault occurs, the system stores a diagnostic trouble code (DTC), also store important information of the vehicle when the fault was set.
- A service technician is able to connect a diagnostic scan tool or a code reader that will communicate with the system and retrieve this information.
- As vehicles and their systems become more complex, the functionality of OBD is being extended to cover vehicle systems and components that do not have anything to do with vehicle emissions control: Vehicle body, chassis and accessories
- OBD systems use a standardized communications port to provide data
- The Communication between Diagnostic Equipment and ECUs through Vehicle Special Interface for Diagnosis purpose is called **Diagnostic Communication.**

Bosch Global Software Technologies
alt_future

Definitions

States and Events

- Everything about the car is either in a "normal" state or an "abnormal" state.
- Either the car is starting normally or it is not. Either the engine is running normally or it is not.
- Used in this way, "normal" means acceptable, the way they are supposed to be, okay. "Abnormal" means not acceptable, not the way they are supposed to be, not okay.
- The purpose of all automobile repair is to correct abnormal states and restore the car to its normal state of operation.



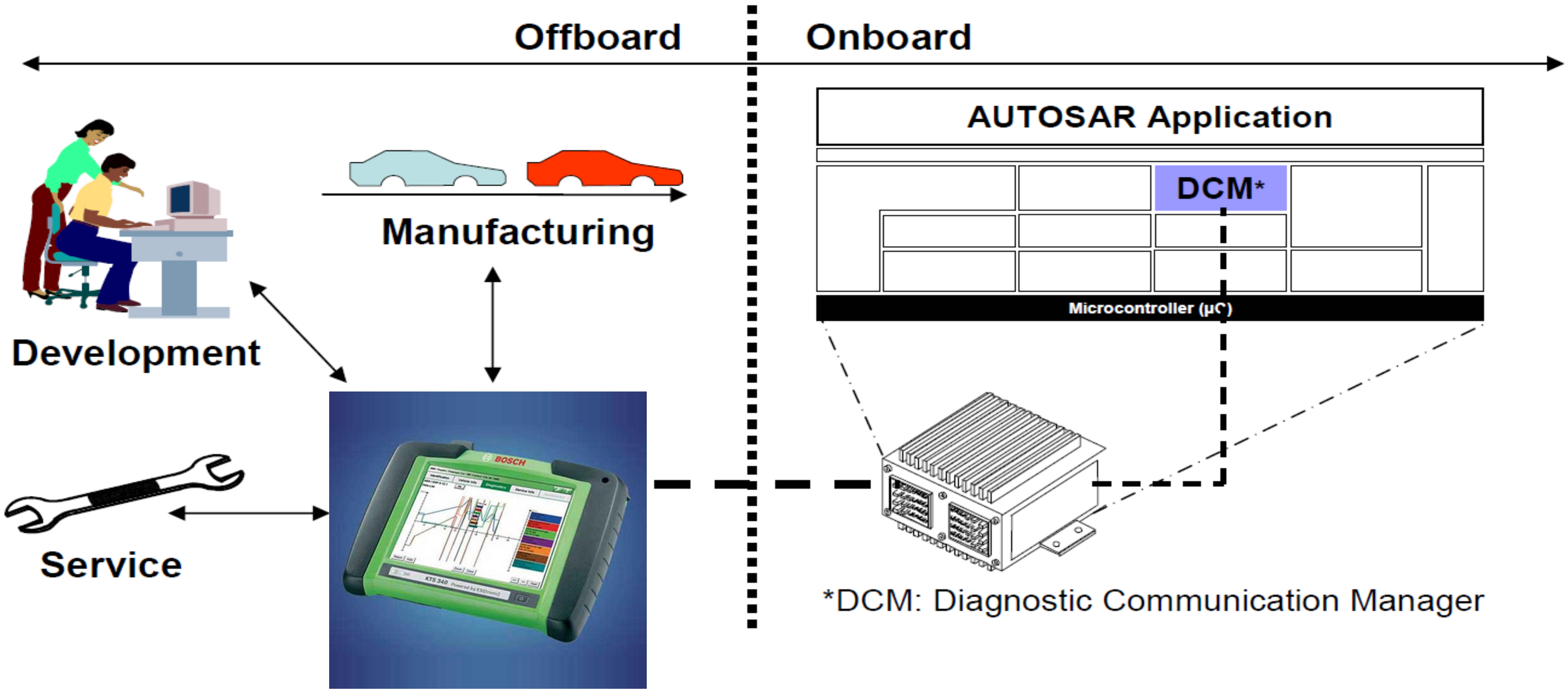
Definitions
States and Events

- An "event" is when something happens: a spark plug fires, the brakes are applied, a fuel injector opens, a relay closes.
- An event is a change of state, from one condition to another.
- A "normal event" occurs when something happens just as it is supposed to happen.
- An "abnormal event" is when something happens that is not supposed to happen: the engine quits unexpectedly, the car fails to stop when the brakes are applied.
- All automotive complaints can be described and understood in terms of abnormal events: some things are happening which are not supposed to happen, and the driver has noticed them.

Diagnosis Uses

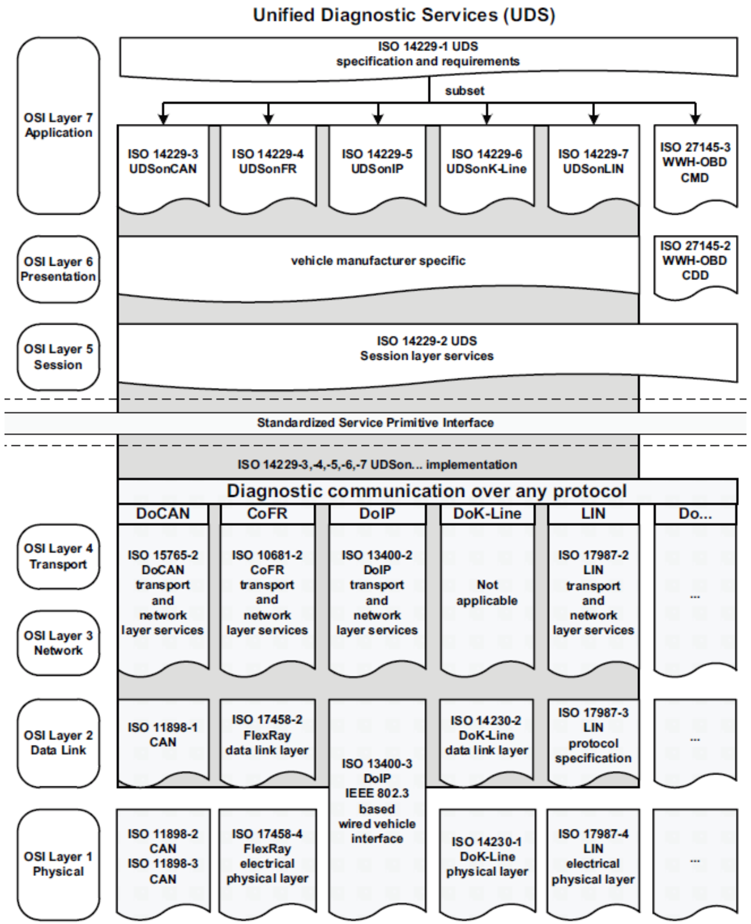
- Diagnosis is used to detect the fault in the system.
- Use to read the parameters like WSS signal, SAS signal, YRS signal etc.
- Used for calibration of Steering Angle sensor, Lateral , Longitudinal sensor etc.
- Use to run EOL(End of Line) routines.
- Used for reprogramming.

On board and Off board Diagnosis



Normally, the information is exchanged between an on-board ECU and an off-board diagnostic tester.

Diagnosis Protocols
Unified Diagnostic Services (UDS)



Bosch Global Software Technologies
alt_future

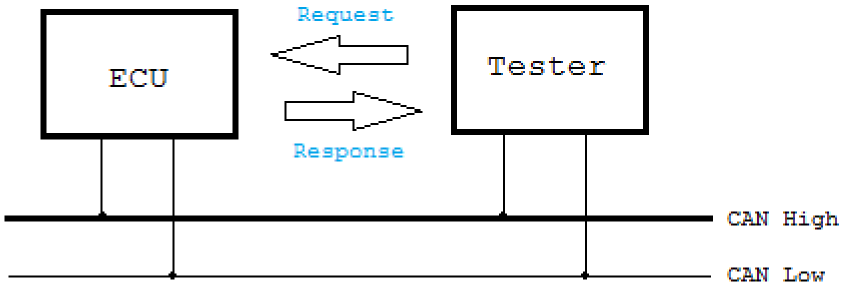
Diagnosis Protocols

Emissions-related diagnostics (emissions-related OBD)

<table>
<tr><th>Applicability</th><th>OSI 7 layers</th><th colspan="5">Emissions-related OBD communication requirements</th><th colspan="3">Emissions-related WWH-OBD communication requirements</th></tr>
<tr><td rowspan="8">Seven layer according to ISO/IEC 7498-1 and ISO/IEC 10731</td><td>Application (layer 7)</td><td colspan="5">ISO 15031-5</td><td colspan="3">ISO 27145-3</td></tr>
<tr><td rowspan="2">Presentation (layer 6)</td><td colspan="5">ISO 15031-2, -5, -6
SAE J1930-DA/SAE J1979-DA</td><td colspan="3">ISO 27145-2
SAE J1930-DA/SAE J1979-DA</td></tr>
<tr><td colspan="5">SAE J2012-DA (OBD)</td><td colspan="3">SAE J2012-DA (WWH-OBD)</td></tr>
<tr><td>Session (layer 5)</td><td colspan="2">Not applicable</td><td colspan="6">ISO 14229-2</td></tr>
<tr><td>Transport (layer 4)</td><td colspan="2" rowspan="2">ISO 15031-5</td><td rowspan="2">ISO 14230-4</td><td rowspan="2">ISO 15765-2</td><td rowspan="4">ISO 15765-4</td><td rowspan="2">ISO 15765-2</td><td rowspan="4">ISO 27145-4</td><td rowspan="2">ISO 13400-2</td></tr>
<tr><td>Network (layer 3)</td></tr>
<tr><td>Data link (layer 2)</td><td rowspan="2">SAE J1850</td><td rowspan="2">ISO 9141-2</td><td>ISO 14230-2</td><td rowspan="2">ISO 11898-1, ISO 11898-2</td><td rowspan="2">ISO 11898-1, ISO 11898-2</td><td rowspan="2">ISO 13400-3</td></tr>
<tr><td>Physical (layer 1)</td><td>ISO 14230-1</td></tr>
</table>

Bosch Global Software Technologies
alt_future

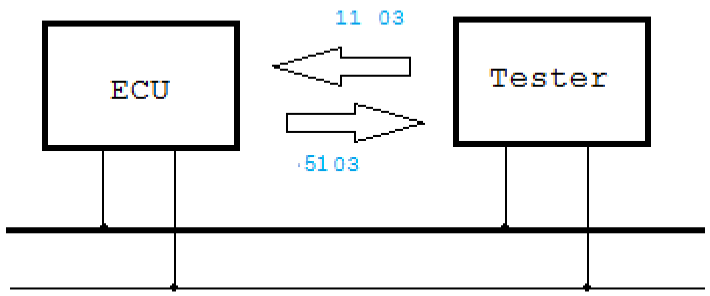
Diagnosis Protocols
ECU(server) – Tester(client) communication



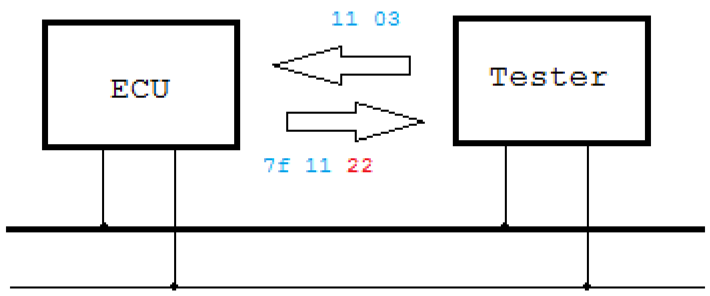
- Tester sends a request\command to ECU, to perform certain action.
- ECU sends a response message to the corresponding request.

Request and response
Overview

Positive response



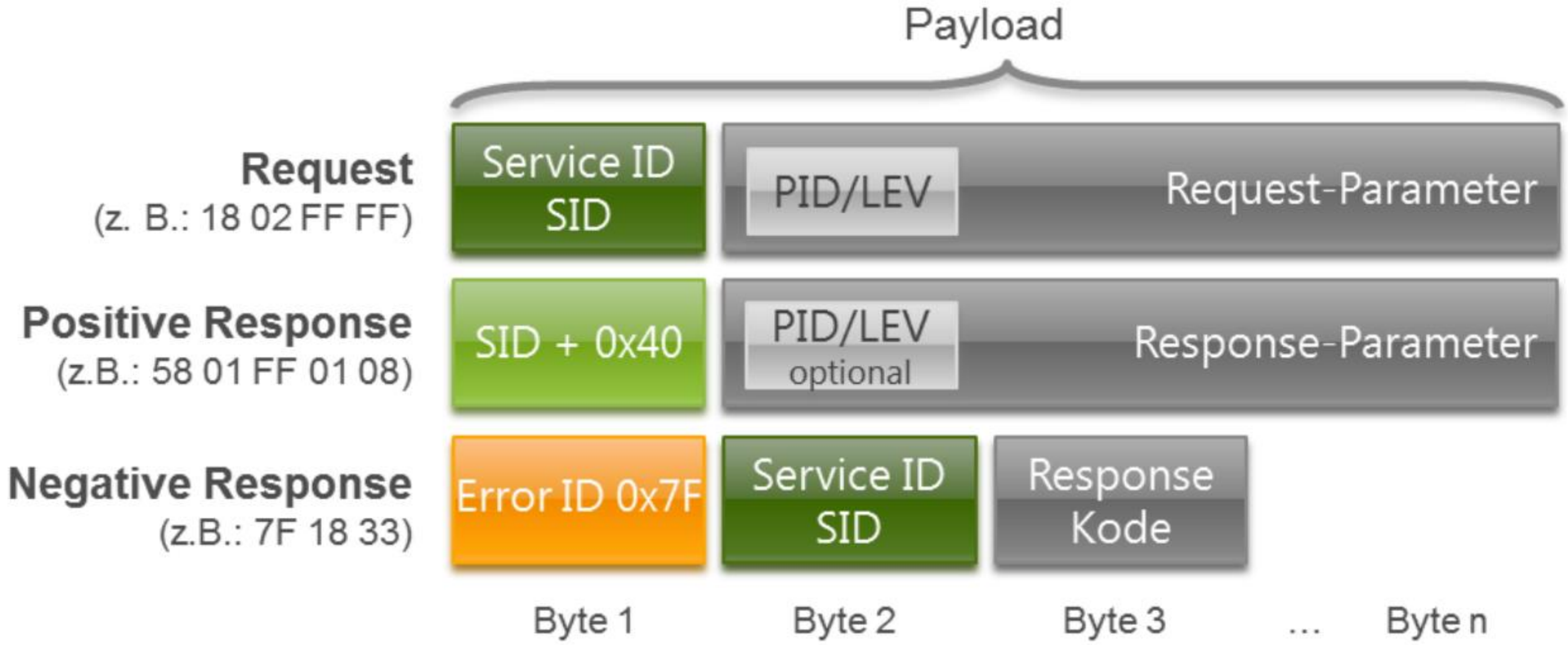
Negative Response



- Tester sends a request to perform software reset.
- ECU returns a positive response.

- Tester sends a request to perform software reset.
- ECU returns a negative response indicating software reset can not be performed.

Request and response
Overview



Bosch Global Software Technologies
alt_future

Request and response
Overview

Request	Positive Response	Description
0x00 .. 0x0F	0x40 .. 0x4F	OBD in ISO 15031-5 Emissions-related diagnostic services
0x10 .. 0x3E 0x83 .. 0x87	0x50 .. 0x7E 0xC3 .. 0xC7	UDS in ISO 14229 KWP2000 in General vehicle diagnostics
0x81.. 0x82	0xC1 .. 0xC2	KWP2000 over K-line in ISO 14230
0xA0 .. 0xB9	0xE0 .. 0xF9	Reverse for OEM
0xBA .. 0xBE	0xFA ..0xFE	Reverse for ECU manufacturers
Others		Reverse

Request and response
Negative response

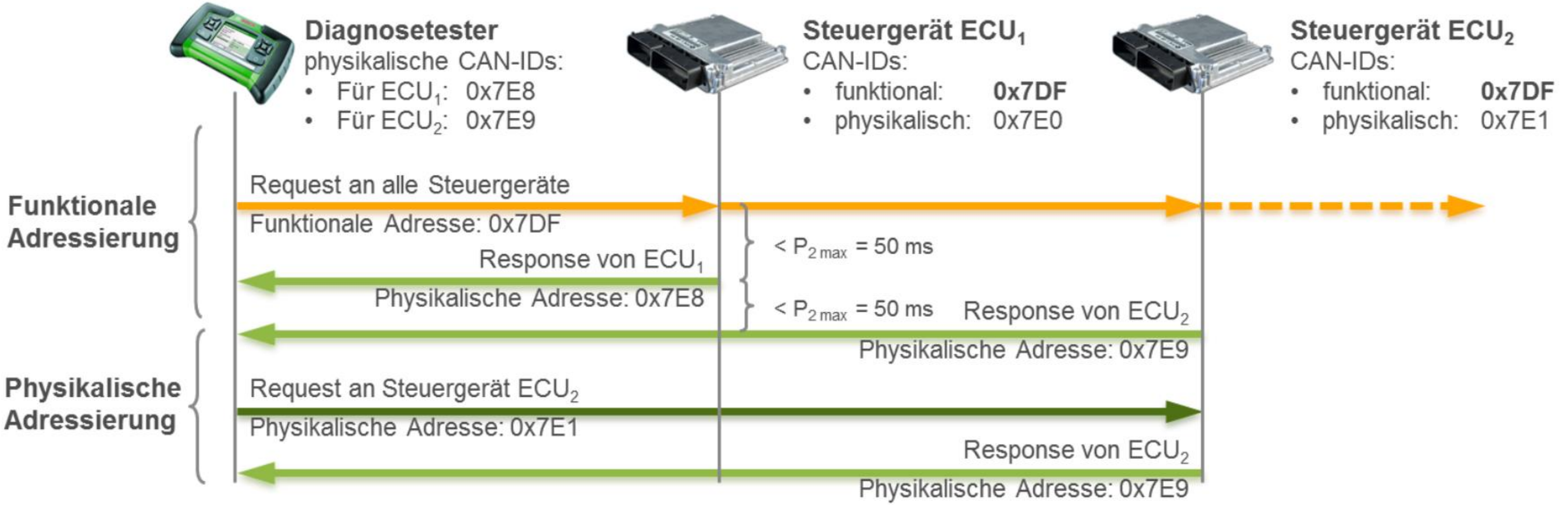
Reponse-Kode	Beschreibung
0x10	General reject
0x11, 0x12, 0x7E, 0x7F	Service or Subfunction not supported (in active Session)
0x13	Message length or format incorrect
0x31	Out of range
0x21	Busy – Repeat request
0x78	Busy – Response pending
0x22	Conditions not correct
0x24	Request sequence error
0x33	Security access denied
0x35	Invalid key
0x36	Exceed attempts

Diagnosis mode

- Normal Mode (Default)
- Test Mode / Adjustment mode
- Reprogramming mode

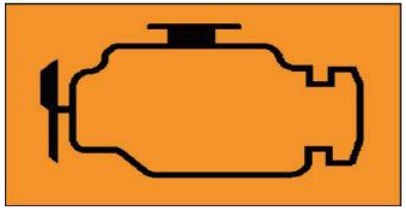
Addressing

Functional Address:
Physical Address:



Bosch Global Software Technologies
alt_future

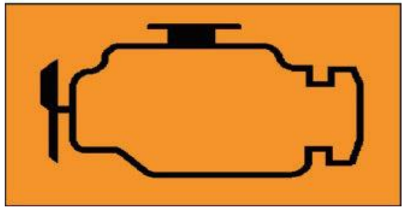
Definitions
Milestones



- **1969:** Volkswagen introduces the first on-board computer system with scanning capability, in their fuel-injected Type 3 models.
- **1975:** Datsun 280Z On-board computers begin appearing on consumer vehicles, largely motivated by their need for real-time tuning of fuel injection systems. Simple OBD implementations appear, though there is no standardization in what is monitored or how it is reported.
- **1980:** General Motors implements a proprietary interface and protocol for testing of the Engine Control Module (ECM) on the vehicle assembly line. The 'assembly line diagnostic link' (ALDL) protocol communicates at 160 baud with Pulse-width modulation (PWM) signalling and monitors very few vehicle systems.

Definitions
Milestones

- **1986:** An upgraded version of the ALDL protocol appears which communicates at 8192 baud with half-duplex UART signalling. This protocol is defined in GM XDE-5024B.
- **~1987:** The California Air Resources Board (CARB) requires that all new vehicles sold in California starting in manufacturer's year 1988 (MY1988) have some basic OBD capability. These requirements are generally referred to as "OBD-I", though this name is not applied until the introduction of OBD-II. The data link connector and its position are not standardized, nor is the data protocol.
- **1988:** The Society of Automotive Engineers (SAE) recommends a standardized diagnostic connector and set of diagnostic test signals.



Definitions
Milestones

- **~1994:** Motivated by a desire for a state-wide emissions testing program, the CARB issues the OBD-II specification and mandates that it be adopted for all cars sold in California starting in model year 1996. The DTCs and connector suggested by the SAE are incorporated into this specification.
- **1996:** The OBD-II specification is made mandatory for all cars sold in the United States.
- **2001:** The European Union makes EOBD mandatory for all gasoline (petrol) vehicles sold in the European Union, starting in MY2001 (see European emission standards Directive 98/69/EC)).

Thank you!

Bosch
Global
Software
Technologies
alt_future