Yrkes Akademin
Vi hjälper dig att lyckas!

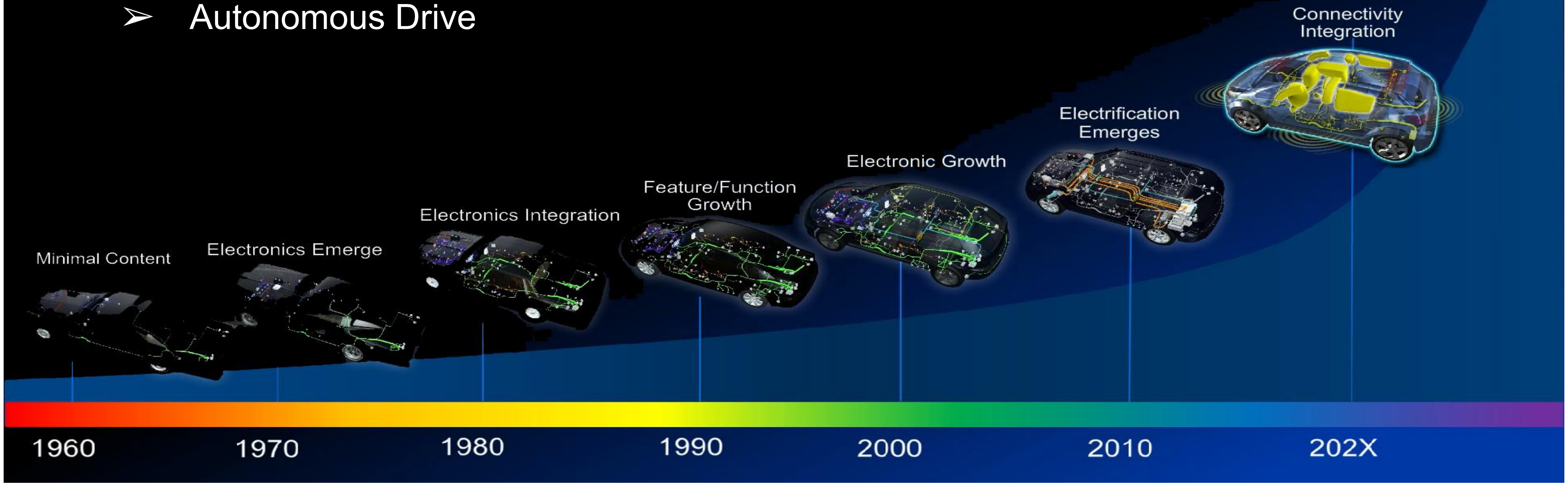
Introduction to Automotive Software Architecture

AUTOSAR

Automotive Software Architecture

- The Big Trends in Automotive
  - Electrification
  - Connectivity
  - Autonomous Drive

[Electrical/Electronic System Evolution]

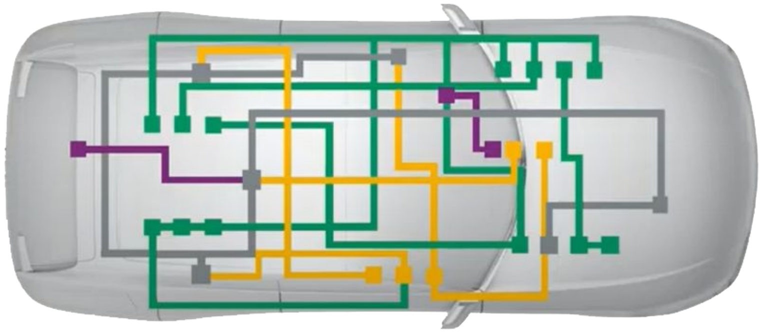


History

- Just 45 years ago first software was deployed into cars
  - Ford EEC was used to control the engine and ignition
  - It was very local and isolated
- Over time different software based functionalities deployed into cars
  - Hardware got cheaper and more powerful
  - ECUs connected to sensors and actuators
  - Bus systems used in order to optimize wiring and exchange data between ECUs
- Nowadays vehicle software based functions are a key differentiator between car brands
  - Premium cars have 70 - 100 ECUs connected by more than 5 different bus systems
  - Up to 40% of the production cost of a car is due to electronics and software

Engineering Automotive Software

Software in Cars

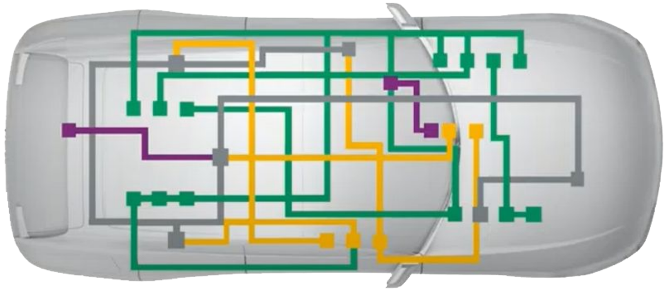


Some Challenges in Software Development for Automotive

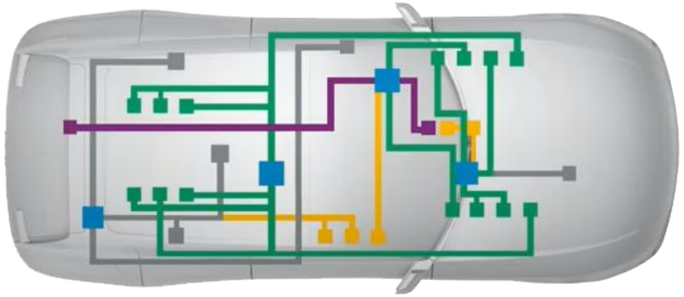
Audi MMI (Man-Machine Interface)

- ❖ Complexity
  - ➢ Not only software - A single car has about 30,000 parts
  - ➢ Gigantic softwares - more than [100 Millions lines of code]
  - ➢ Different architectures of more than 100 ECU hardwares
    - ■ Distributed, Domain and Central
  - ➢ Many standards to follow.
    - ■ E.g. AutoSar, ISO 26262, MISRA C and etc.

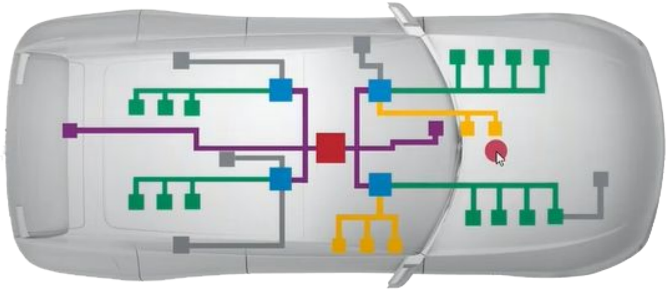
Images from: AUTOSAR Basic Software and Beyond



Distributed Architecture



Domain Architecture

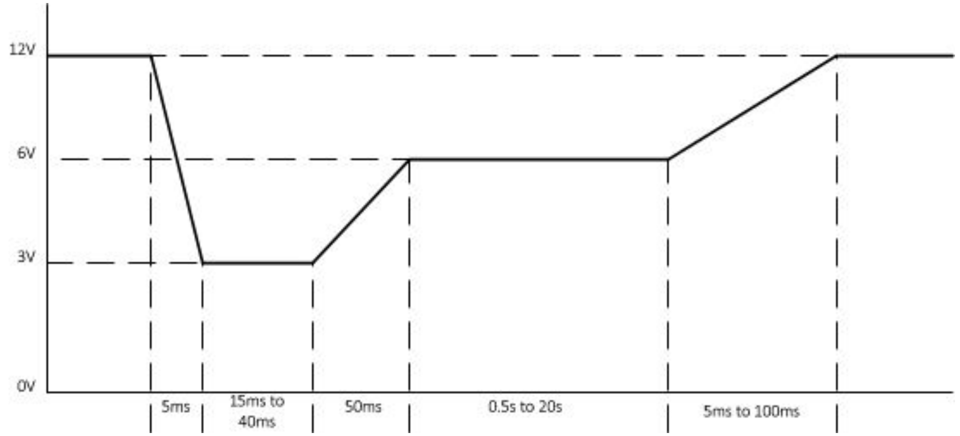


Central Architecture

Yrkes Akademin
Vi hjälper dig att lyckas!

Some Challenges in Software Development for Automotive

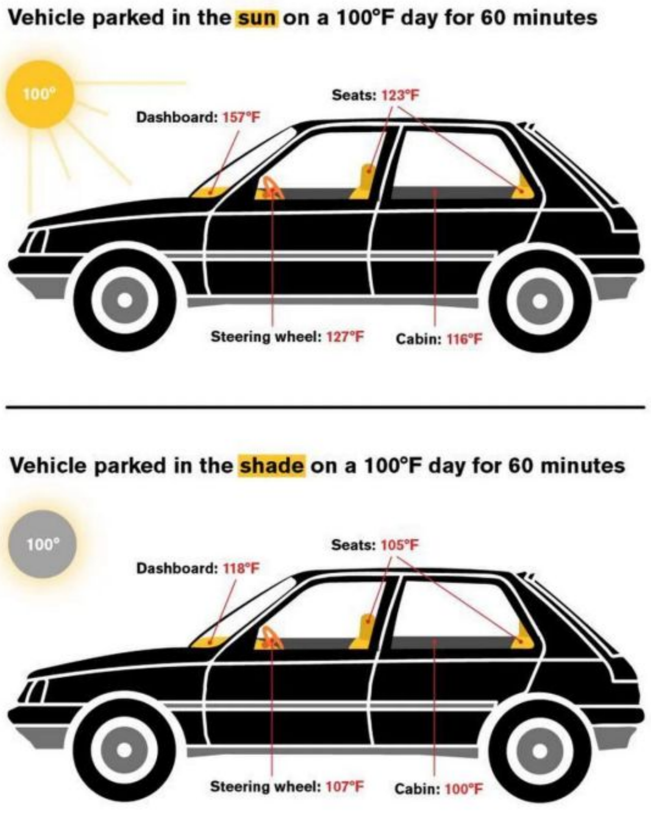
- Voltage drops of the battery
  - E.g. When starting the engine in the morning and even worse in cold weather
  - You can not rely on unstable and low voltage to the electronics
  - E.g. Software writing to EEPROM memory can not be reliable when power is low
- ECUs supporting CAN will be notified a temporary low battery voltage condition will occur
- Software needs gracefully handle power drops
  - Store current state just before low battery voltage condition occurrence
  - Once the Battery voltage returns to nominal (e.g. 12.8 V) the controllers should safely resume its operation



Some Challenges in Software Development for Automotive

- ❖ Electronic components operate in different temperature range
  - ➢ Commercial: 0 °C to 85 °C
  - ➢ Industrial: −40 °C to 100 °C
  - ➢ Automotive: −40 °C to 125 °C
  - ➢ Military: −55 °C to 125 °C

100 °F = 37.8 °C
157 °F = 70 °C



Data Source: Vanos, Middel, Poletti, Selover. *Temperature*, 2018.

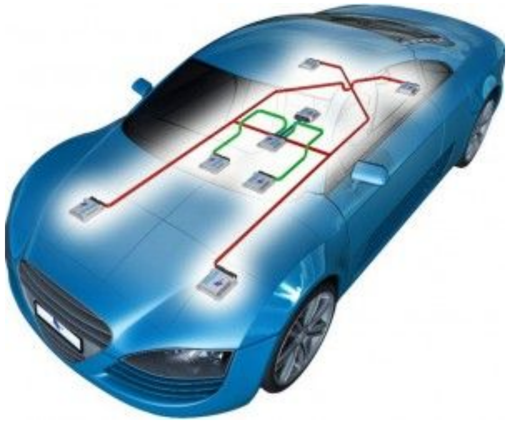
- ❖ Software need to gracefully handle overheated electronics

- ❖ It can be cold!
- ❖ Car displays does not work well in -40 °C

Some Challenges in Software Development for Automotive

- Production Cost
    - When building millions of cars, cost per unit must be kept low

- As a software developer you will always have
    - Limited processing power
    - Limited memory
    - Limited graphics performance
    - Low data rates
    - This can be a really challenging

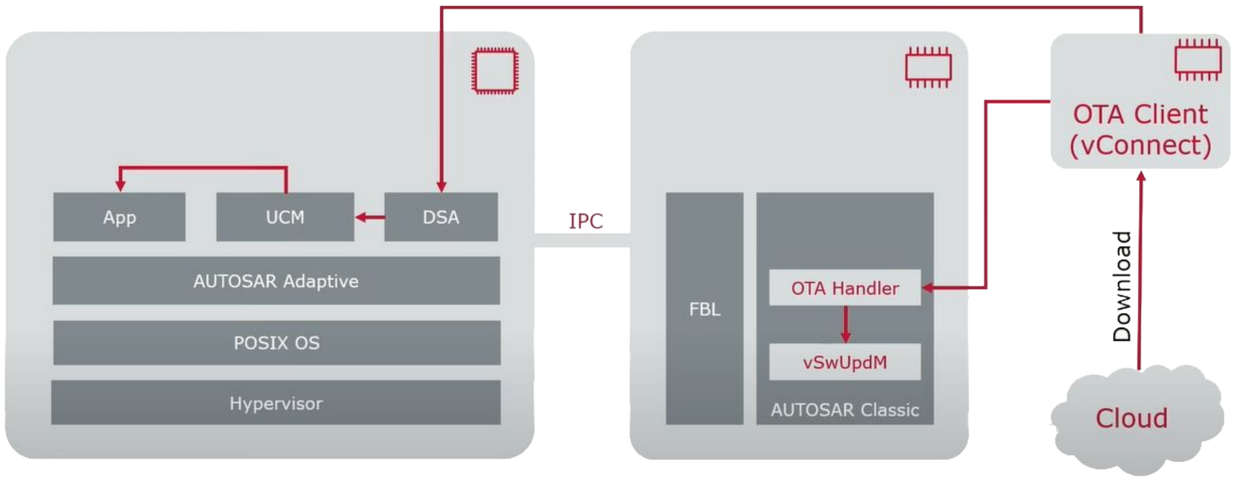


[Some Used Microcontrollers in Automotive Application]

- E.g. ATmega32; 8 bit, 1 MHz, 32K Flash Memory and 2K bytes of internal SRAM

Some Challenges in Software Development for Automotive

- Update software over the air (OTA)
  - When is it safe to update the software?
    - When driving - No
    - When car is stopped - Maybe



Tesla

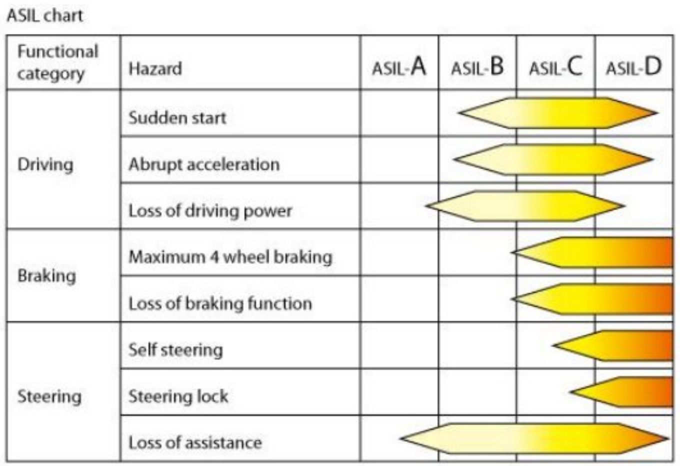
Some Challenges in Software Development for Automotive

- Security
  - Connected vehicles are attractive targets for hackers
  - Similar to the first computers connecting to the Internet, current automotive architectures have not been designed with respect to security, making them highly vulnerable.

[Hackers Remotely Kill a Jeep on the Highway]

- Safety - A vehicle is a safety-critical system
  - How can you write safe software and handle failures?
  - Some safety domains in a car
    - Braking system, Steering system, MMI and etc.
    - Obstacle detection and avoidance in self-driving cars
    - And etc.

ISO 26262 defines functional safety for automotive equipment



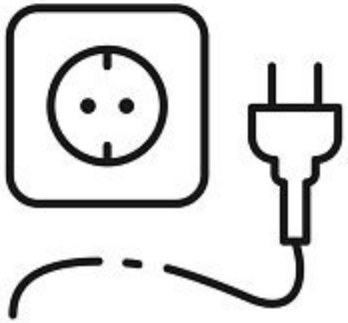
Some Challenges in Software Development for Automotive

- Testing and Validation - every single functionality shall be tested
  - Is the function correct? Are the integrated subsystem and system correct?
  - Is the timing correct? Many hard real-time systems; E.g. airbag, emergency braking system
  - How can we verify a 100 millions line of code system?
- Supply Chain: Many players involved in the software development
  - Manufacturers (OEM)
    - Automotive manufacturers like Ford and Toyota and BMW and etc.
  - Tier 1 Suppliers
    - Companies that supply parts or systems directly to OEMs. E.g. Bosch, Continental and etc.
  - Tier 2 Suppliers
    - Many firms supply parts used in cars, even if they do not sell directly to OEMs.
    - E.g. Semiconductor manufacturers like NXP, Intel or NVIDIA.

Automotive Software Architecture

- ❖ Why can we power different devices by plugging them to el.?
  - ➢ Standard and fixed interfaces
  - ➢ Abstraction

A module is a self-contained and encapsulated unit of a software system which provides some functionalities by having a well-defined interface and an implementation which hides the details

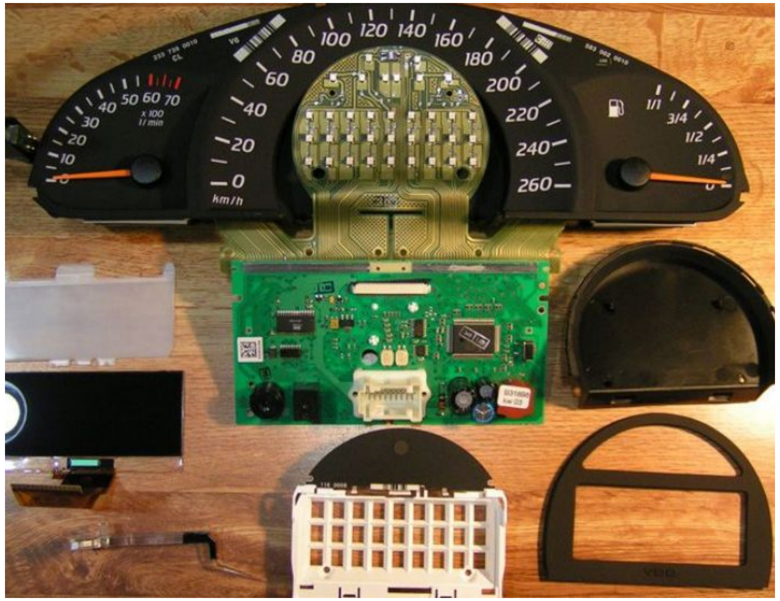


El. Plug & Socket

The buttons of a TV controller are the interfaces to different functionalities

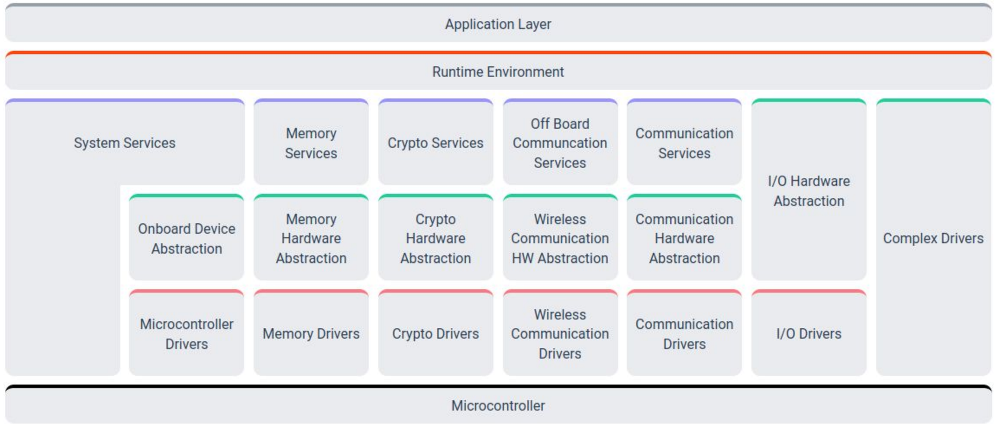
Automotive Software Architecture

- Why a standard architecture?
  - Complex software
  - Many OEMs and Tier 1s are involved
  - Make reusable, maintainable, portable and hardware independent applications
  - Isolate modules and focus on functionalities
- Typical functionalities of an ECU
  - Input/Output
  - Communication (CAN, LIN, and etc.)
  - Timing and scheduling
  - Display & HMI (Human-Machine Interface)
  - Customization
  - Diagnostics
  - And etc.



Automotive Software Architecture - AUTOSAR

- AUTOSAR (AUTomotive Open System ARchitecture)
  - Industry standard for automotive SW architectures
  - Developed thru cooperation between most major car and truck OEMs and Tier 1’s
  - Created to enable reuse, expand supplier base (for BSW) and standardize the industry
  - Very complex solution. Only a small part is used in a specific project
  - Supports all active technologies
  - Support for functional safety
  - Classic and Adaptive platforms
    - We focus on the classic platform
  - Is dedicated for Automotive ECUs
    - Not the mechanical design

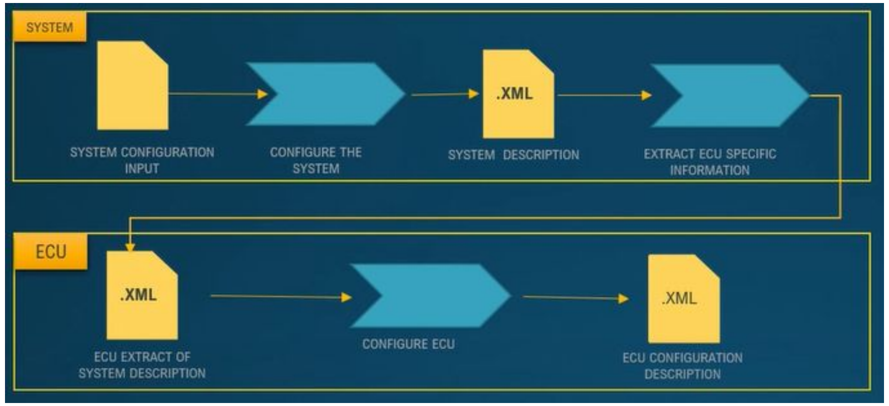


ECU: A microcontroller and connected peripherals according to the software/configuration

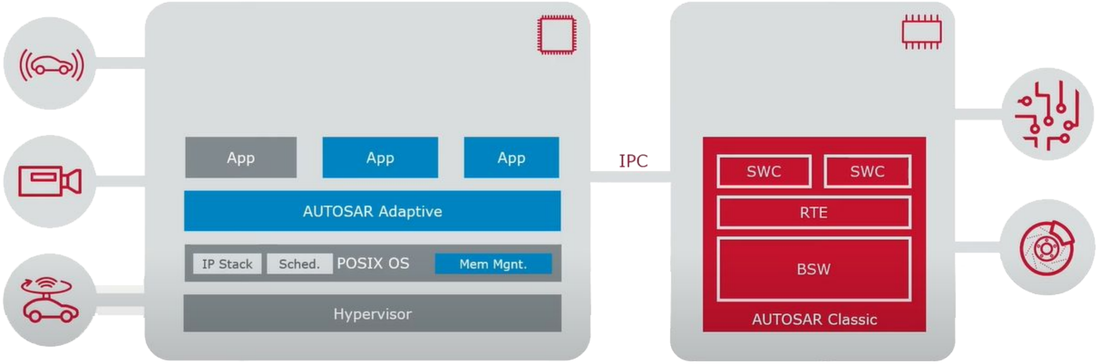
An example of the classic AUTOSAR

Automotive Software Architecture - AUTOSAR

- ❖ Standard and fixed interfaces and abstractions are critical
- ❖ AUTOSAR defines standard XML based templates(ARXML) in order to exchange data and descriptions files of system, components and resources between the participating members.



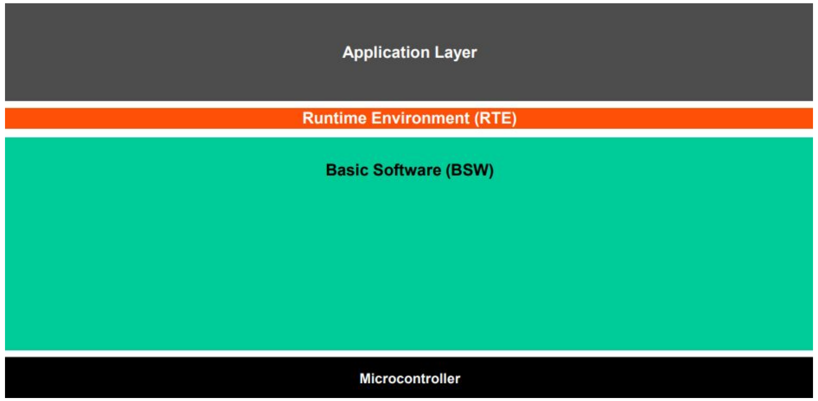
AUTOSAR workflow and ARXMLs

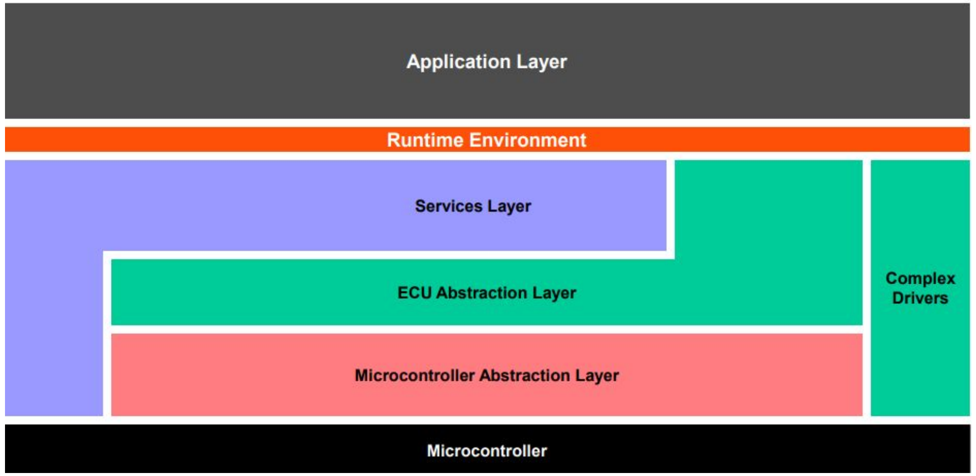


Classic and Adaptive AUTOSAR

Automotive Software Architecture - AUTOSAR

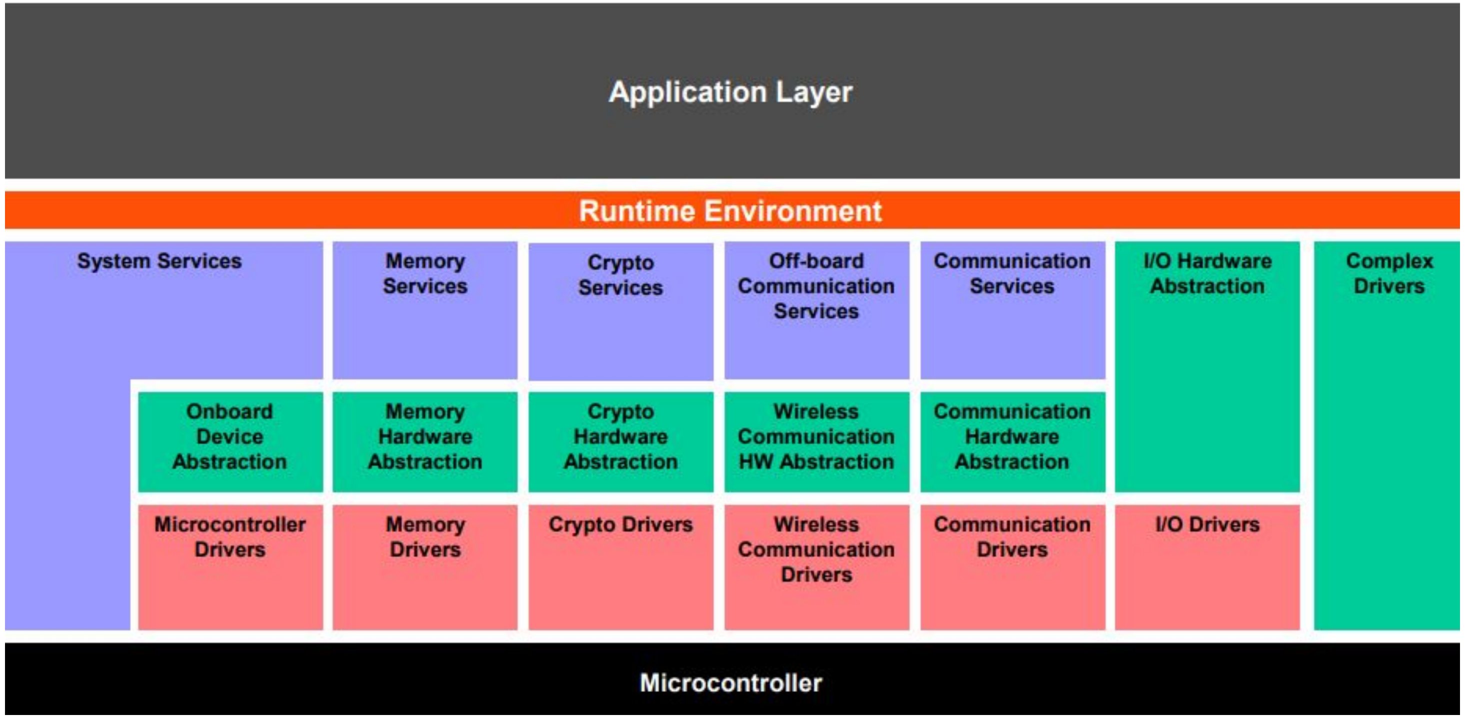
- The AUTOSAR abstracts hardware components from the application layer
- The AUTOSAR is a layered architecture
- The AUTOSAR highest abstraction level
  - Application Layer
  - Runtime Environment
  - Basic Software which is divided to:
    - Microcontroller Abstraction Layer
    - ECU Abstraction Layer
    - Services Layer
    - Complex Drivers





Automotive Software Architecture - AUTOSAR

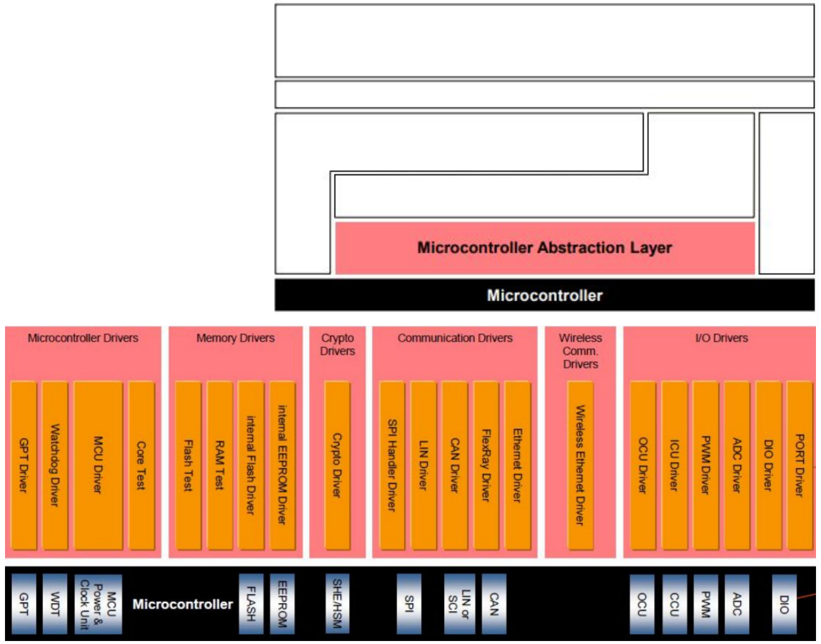
- The Basic Software Layers are further divided into functional groups and modules
  - E.g. Drivers, hardware abstractions, memory and communication services and etc.



A typical Basic Software architecture and the functional groups that are usually present

Automotive Software Architecture - AUTOSAR

- ❖ Microcontroller Abstraction Layer
  - ➢ Different names: BSP / MCAL
  - ➢ The first layer in the abstraction
  - ➢ It contains internal drivers, which are software modules with direct access to the µC and internal peripherals.
  - ➢ Makes higher software layers independent of µC
  - ➢ The implementation is µC dependent and the upper interface is standardized and µC independent
  - ➢ Examples:
    - ■ Memory/IO connected via I2C/SPI is abstracted in the BSP
    - ■ Mode management: ECU_Sleep, ECU_Standby and etc.
    - ■ IO functionality e.g. DIO_ReadChannel

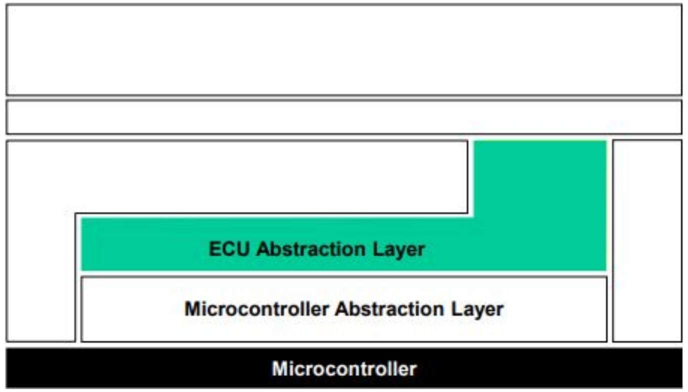


SWS_Dio_00133] [

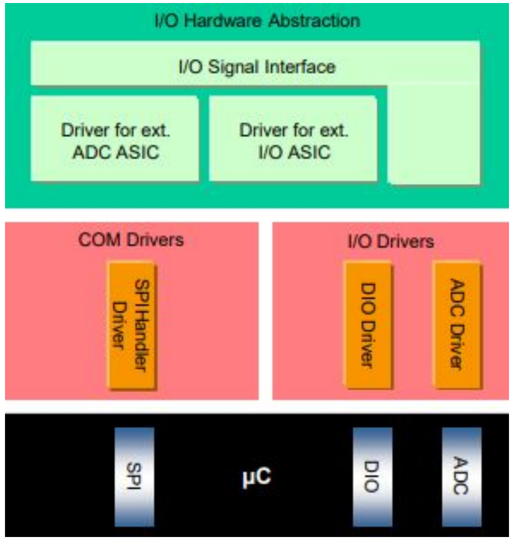
Service name:	Dio_ReadChannel	
Syntax:	`Dio_LevelType Dio_ReadChannel( Dio_ChannelType ChannelId )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	ChannelId	ID of DIO channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Dio_LevelType	STD_HIGH The physical level of the corresponding Pin is STD_HIGH STD_LOW The physical level of the corresponding Pin is

Automotive Software Architecture - AUTOSAR

- ECU Abstraction Layer
  - Interfaces the drivers of the MCAL.
  - 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)
  - Makes higher software layers independent of ECU hardware
  - The implementation is µC independent and ECU dependent
  - The upper interface is µC and ECU hardware independent
  - Example: IO Hardware Abstraction
    - Second layer of the HW detachment
    - Abstracts from functional names to actual IO ports
    - E.g. Set_BrkLight_Ind(HIGH) -> BSP_SetDO_2(HIGH)

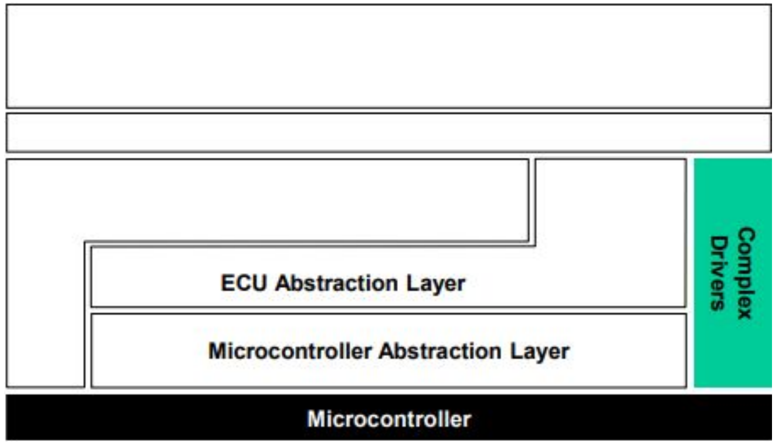


Example



Automotive Software Architecture - AUTOSAR

- Complex Drivers Layer
  - Provides the possibility to integrate special purpose functionality
  - E.g. drivers for devices which:
    - Are non-standard and not specified within AUTOSAR
    - Are hardware dependent and can not be run in application space; E.g. using interrupts
    - Are timing critical and have very high timing constraints
  - The implementation and the upper Interface
    - Might be application, µC and ECU hardware dependent
    - But the upper interface fulfills the standard Autosar
  - Examples
    - Injection control
    - Electric valve control
    - Incremental position detection



Automotive Software Architecture - AUTOSAR

❖ Services Layer

➢ Is the highest layer of the Basic Software

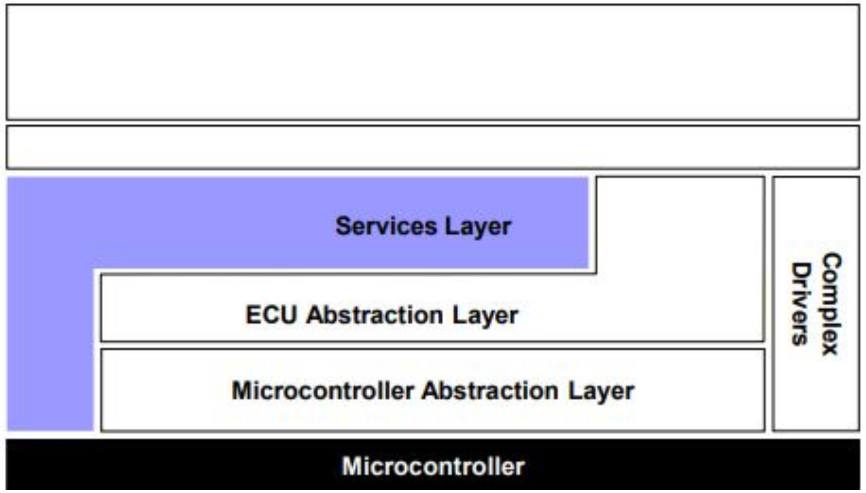
➢ Offers all the services needed in the platform

- Operating system functionality
- Memory services (NVRAM management)
- ECU state and mode managements
- Watchdog management
- Vehicle network communication and management services
- Diagnostic Services. E.g. Unified Diagnostic Services (UDS)

➢ The implementation is mostly µC and ECU hardware independent

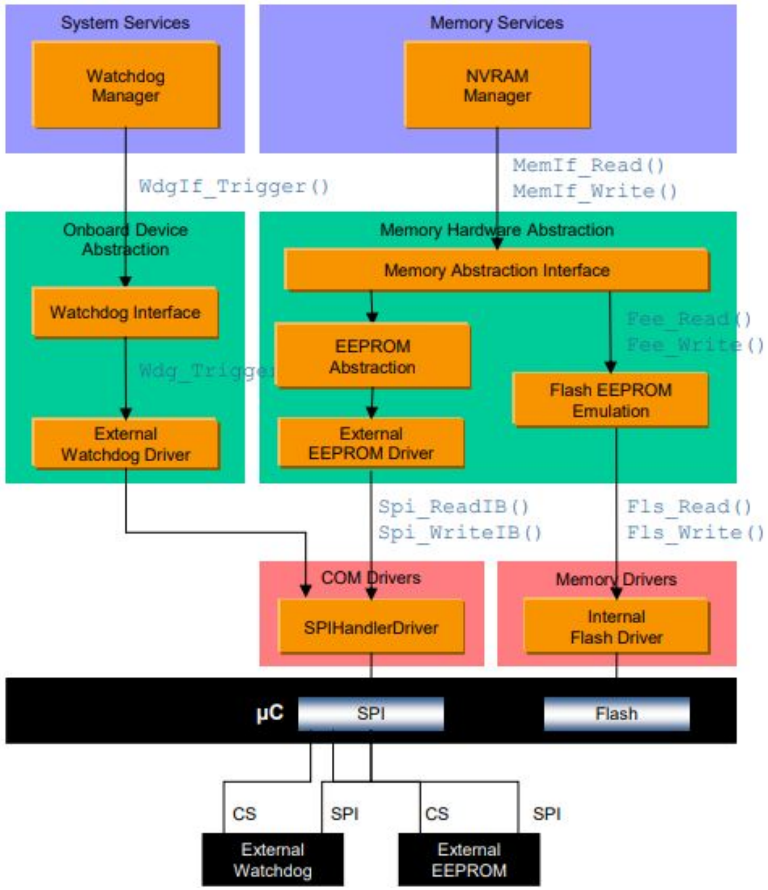
➢ The upper Interface is µC and ECU hardware independent

➢ Example: Memory Services



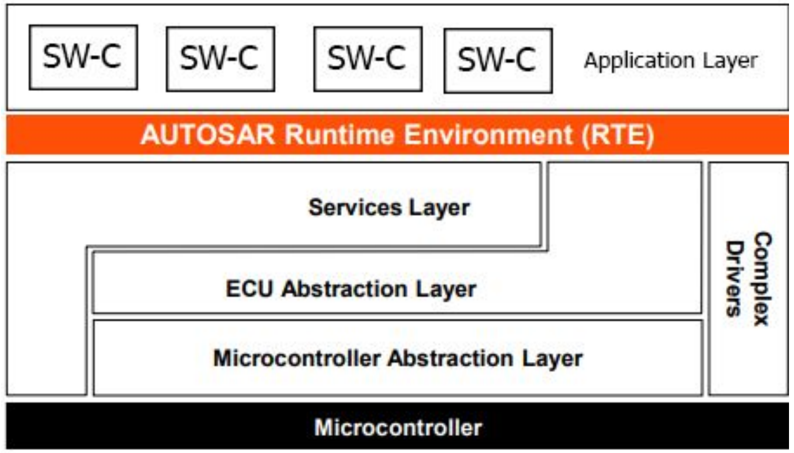
Automotive Software Architecture - AUTOSAR

- Example: Memory Services
  - Handles Non-volatile memory access
  - Maps storage identifiers to memory access using a device index
  - Loads calibration parameters at startup
  - Handles buffering etc.
  - Storage can be one memory type or many different memory locations & types
  - Abstracts the actual NV storage and provides standard interfaces for the upper layers
    - E.g. Nvm_Write(BlockIndex) and Nvm_Read(BlockIndex)



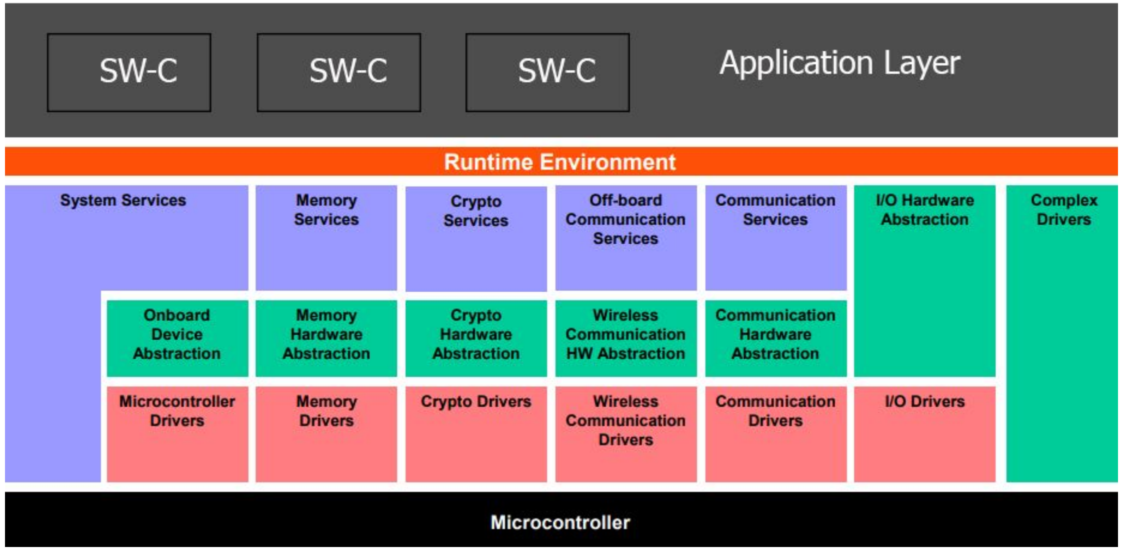
Automotive Software Architecture - AUTOSAR

- Runtime Environment (RTE)
  - Makes AUTOSAR Software Components independent from the mapping to a specific ECU.
  - Provides stable and fixed interface for the applications
    - Allows reuse of applications between projects
  - Provides interaction between software components
  - Interfaces defined in ports and interfaces
  - Interface towards the SW-Cs:
    - Communication, Scheduling, Calibrations, and etc.
  - The implementation is ECU and application specific
    - Generated individually for each ECU
  - The upper interface is completely ECU independent



Automotive Software Architecture - AUTOSAR

- Application Layer
  - Completely independent of a specific ECU
  - Business logic & value
  - Time triggered
  - Execution requirements
  - Described in a fixed format
  - Required interfaces & signals
  - Calibration and storage requirements
  - And etc.



  - Example: The Sensor/Actuator AUTOSAR Software Component

Automotive Software Architecture

❖ Some useful links

- ➢ AUTOSAR Classic Platform
- ➢ AUTOSAR Adaptive Platform
- ➢ AUTOSAR (Automotive Open System Architecture)
- ➢ AUTOSAR Explained Layered Software Architecture
- ➢ AUTOSAR Basic Software and Beyond
- ➢ Introduction to AUTOSAR