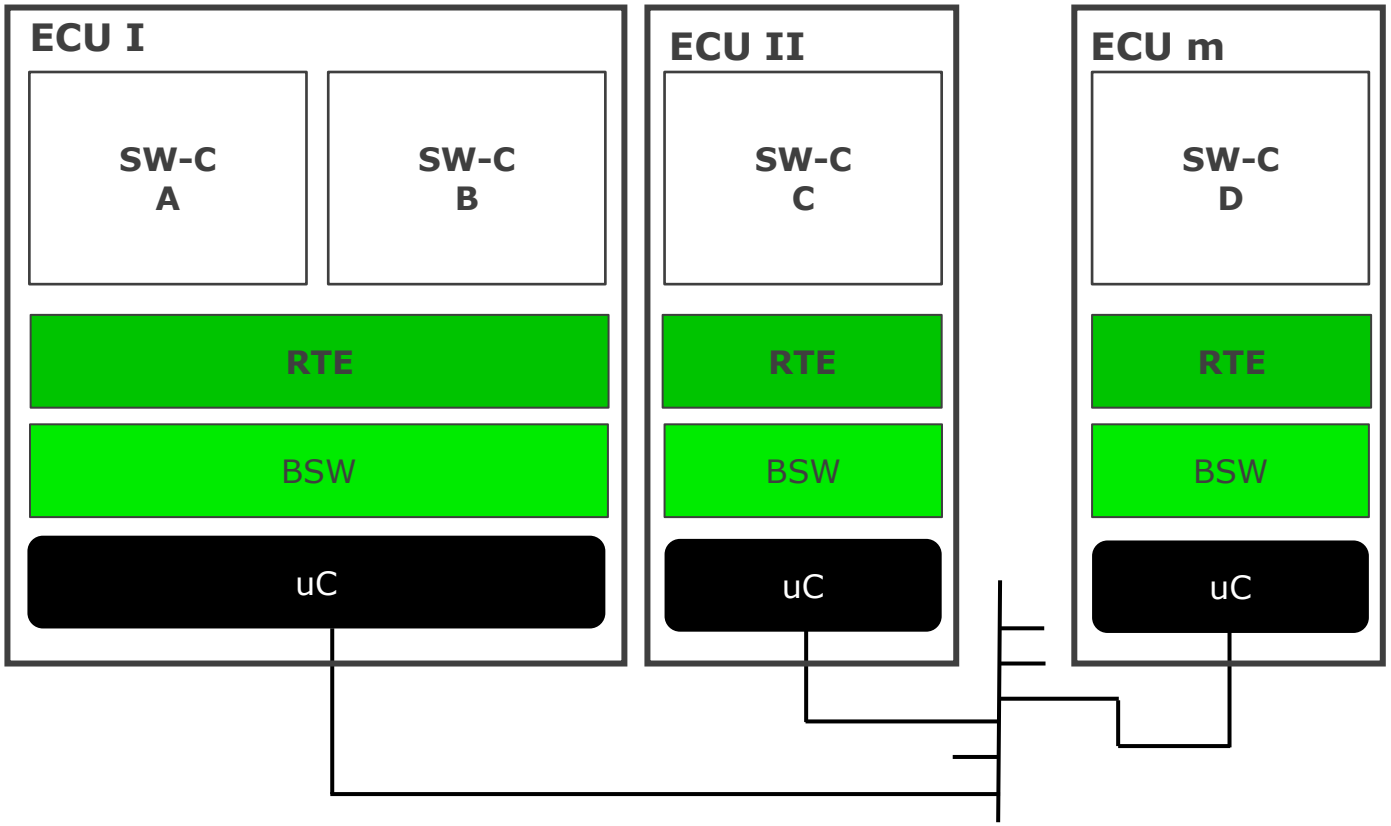
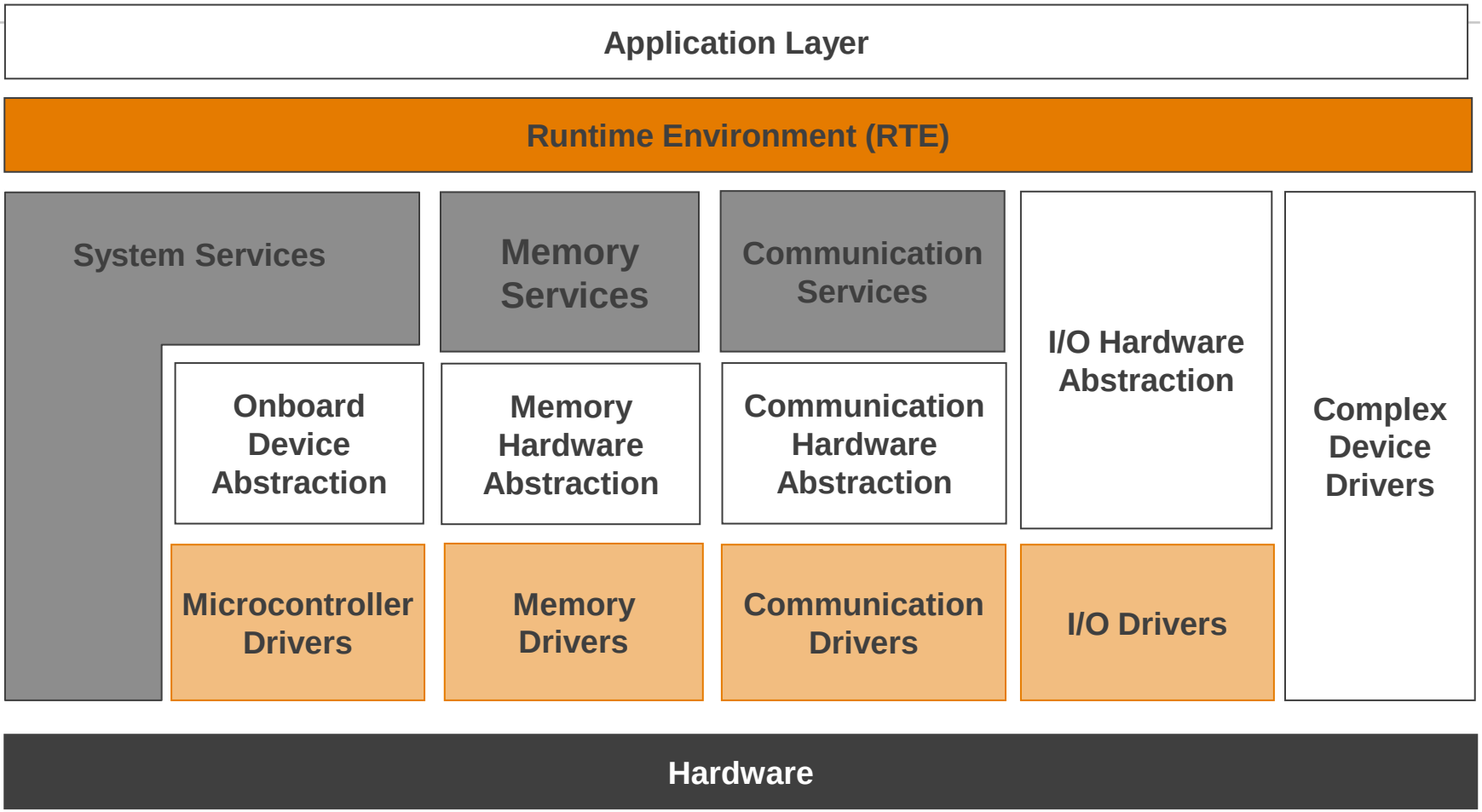
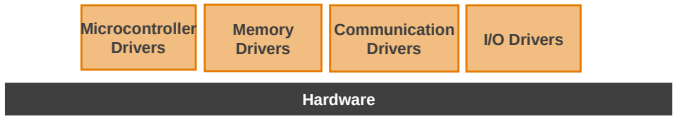
EB tresos classic AUTOSAR training - Architecture

Elektrobit



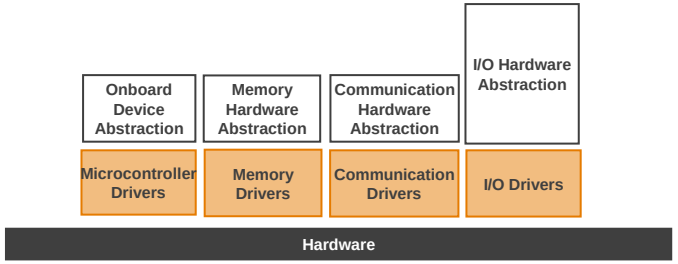


- The ***Microcontroller Abstraction Layer*** is the lowest software layer of the Basic Software. It contains internal drivers, which are software modules with direct access to the µC internal peripherals and memory mapped µC external devices.



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

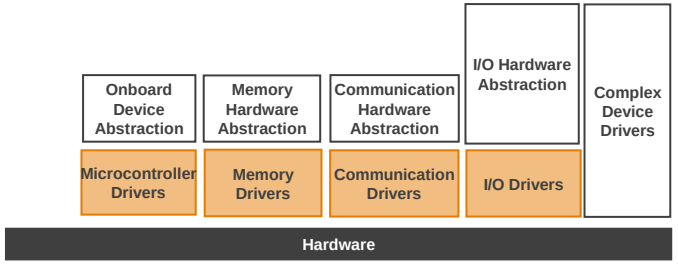
- The ***ECU Abstraction Layer*** interfaces the drivers of the Microcontroller Abstraction Layer. It also contains drivers for external devices. It offers an API for access to peripherals and devices regardless of their location (µC internal/external) and their connection to the µC (port pins, type of interface)



- Task:
  - Make higher software layers independent of ECU hardware layout, e.g. bus types, memory devices

- Properties:
  - Implementation: µC independent, ECU hardware dependent
  - Upper Interface: µC and ECU hardware independent, dependent on signal type

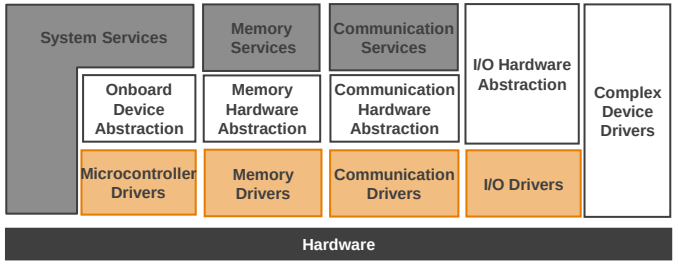
- A ***Complex Device Driver*** implements complex sensor evaluation and actuator control with direct access to the µC using specific interrupts and/or complex µC peripherals (like PCP, TPU), e.g.
  - Injection control
  - Electric valve control
  - Incremental position detection



- Task:
  - Fulfill the special functional and timing requirements for handling complex sensors and actuators

- Properties:
  - Implementation: Highly µC, ECU and application dependent
  - Upper Interface: Specified and implemented according to AUTOSAR (AUTOSAR interface)

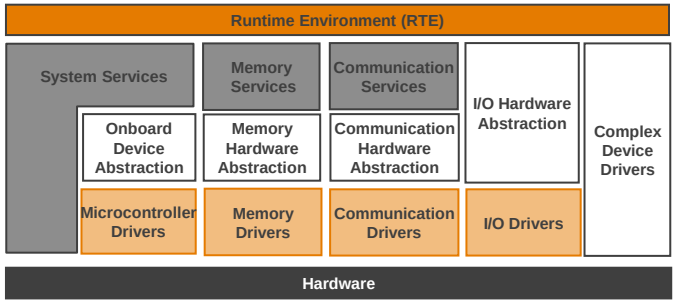
- The ***Service Layer*** is the highest layer of the Basic Software which also applies for its relevance for the application software: while access to I/O signals is covered by the ECU Abstraction Layer, the Services Layer offers:
  - Operating system functionality
  - Vehicle network communication /management
  - Memory services (NVRAM management)
  - Diagnostic Services (UDS, OBD)
  - Mode management



- Task:
  - Provide basic services for application and basic software modules.

- Properties:
  - Implementation: Partly µC, ECU hardware and application specific
  - Upper Interface: µC and ECU hardware independent

- 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.

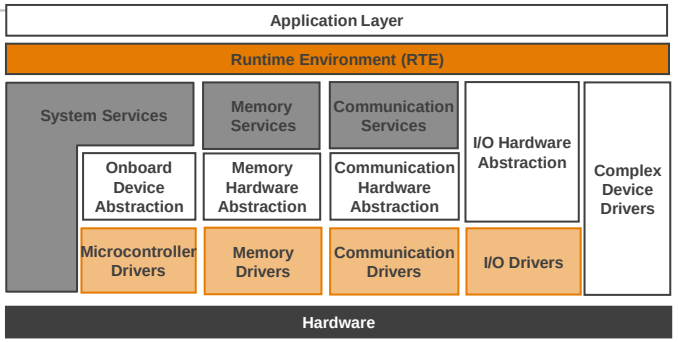


- Task:
  - Make AUTOSAR Software Components independent from the mapping to a specific ECU

- Properties:
  - Implementation: ECU and application specific (generated individually for each ECU)
  - Upper Interface: Completely ECU independent

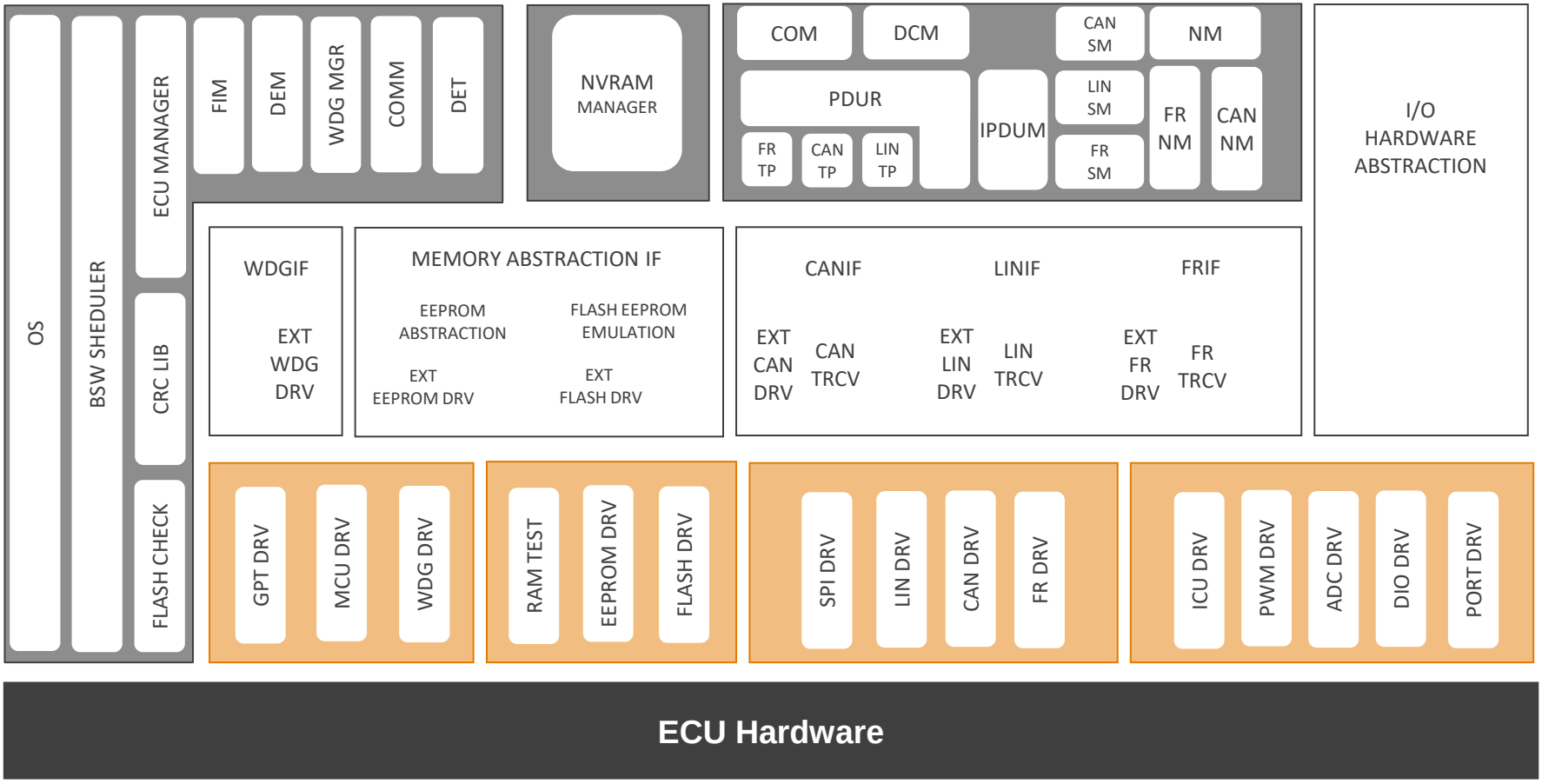
- The ***Application Layer*** is a layer providing application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components).

  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.

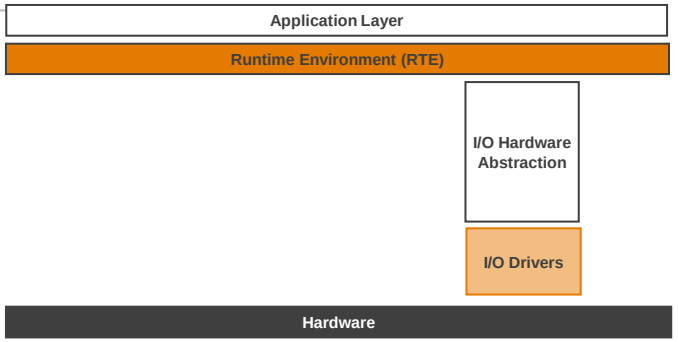


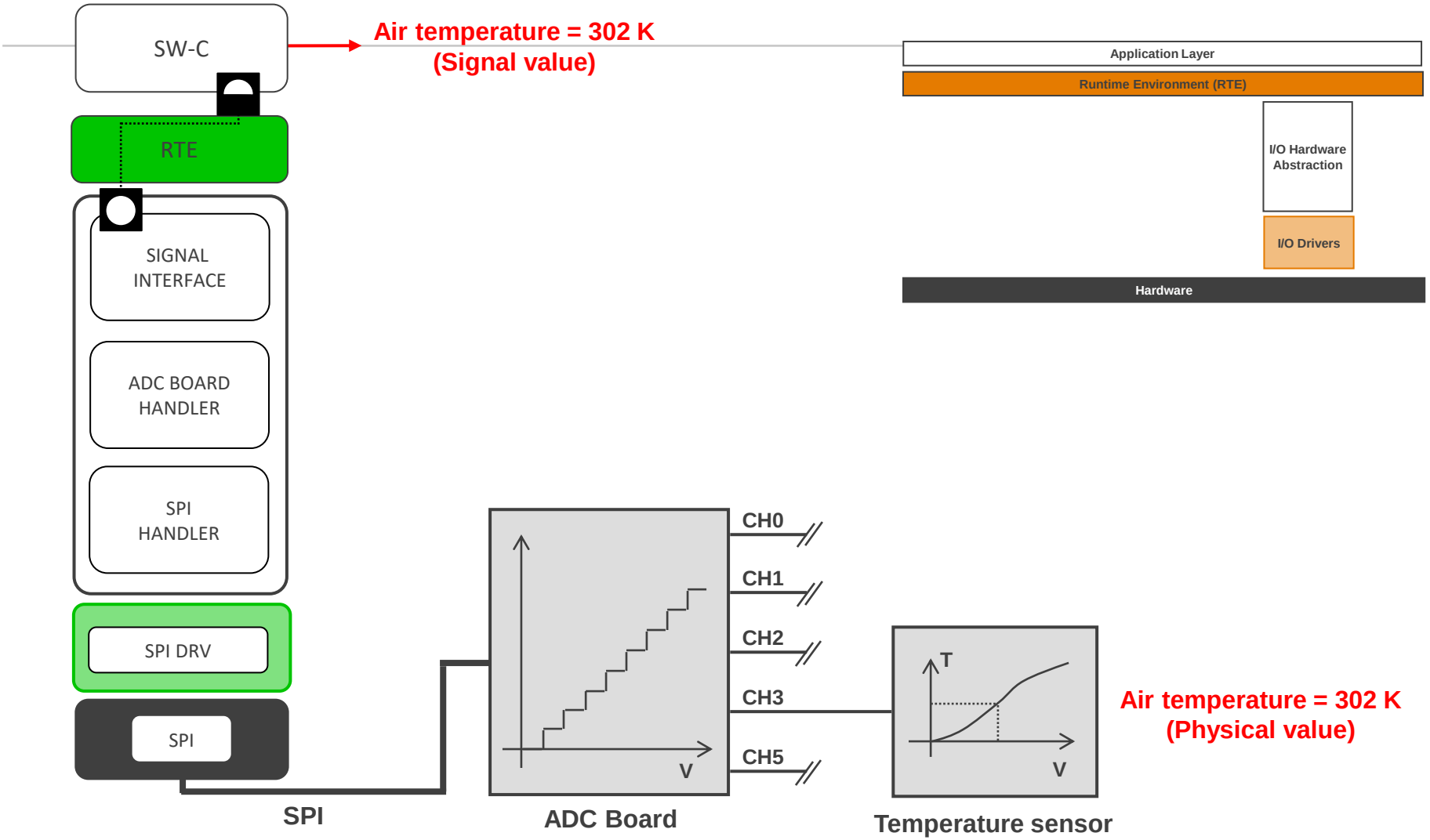
- Task:
  - Implement applications (runnables) that are executed by the RTE

- Properties:
  - Applications completely ECU independent.
  - Sensor/Actuator SW-Cs are dependent on the specifics of a sensor or actuator.



- Project specific
  - AUTOSAR provides high level requirements and guidelines
- Signal based interface
  - Digital IO
  - Analogue IO
  - PWM
- Encapsulates
  - Mapping signal to pin
  - Filtering and de-bouncing
  - Failure monitoring (SC to ground/power, open load, …) and reporting
  - Compensation of static influences
  - Conversation to physical units
  - Handling of SPI driven devices




SW-C
Air temperature = 302 K
(Signal value)
RTE
SIGNAL INTERFACE
ADC BOARD HANDLER
SPI HANDLER
SPI DRV
SPI
Application Layer
Runtime Environment (RTE)
I/O Hardware Abstraction
I/O Drivers
Hardware
CH0
CH1
CH2
CH3
CH5
V
T
V
SPI
ADC Board
Temperature sensor
Air temperature = 302 K
(Physical value)

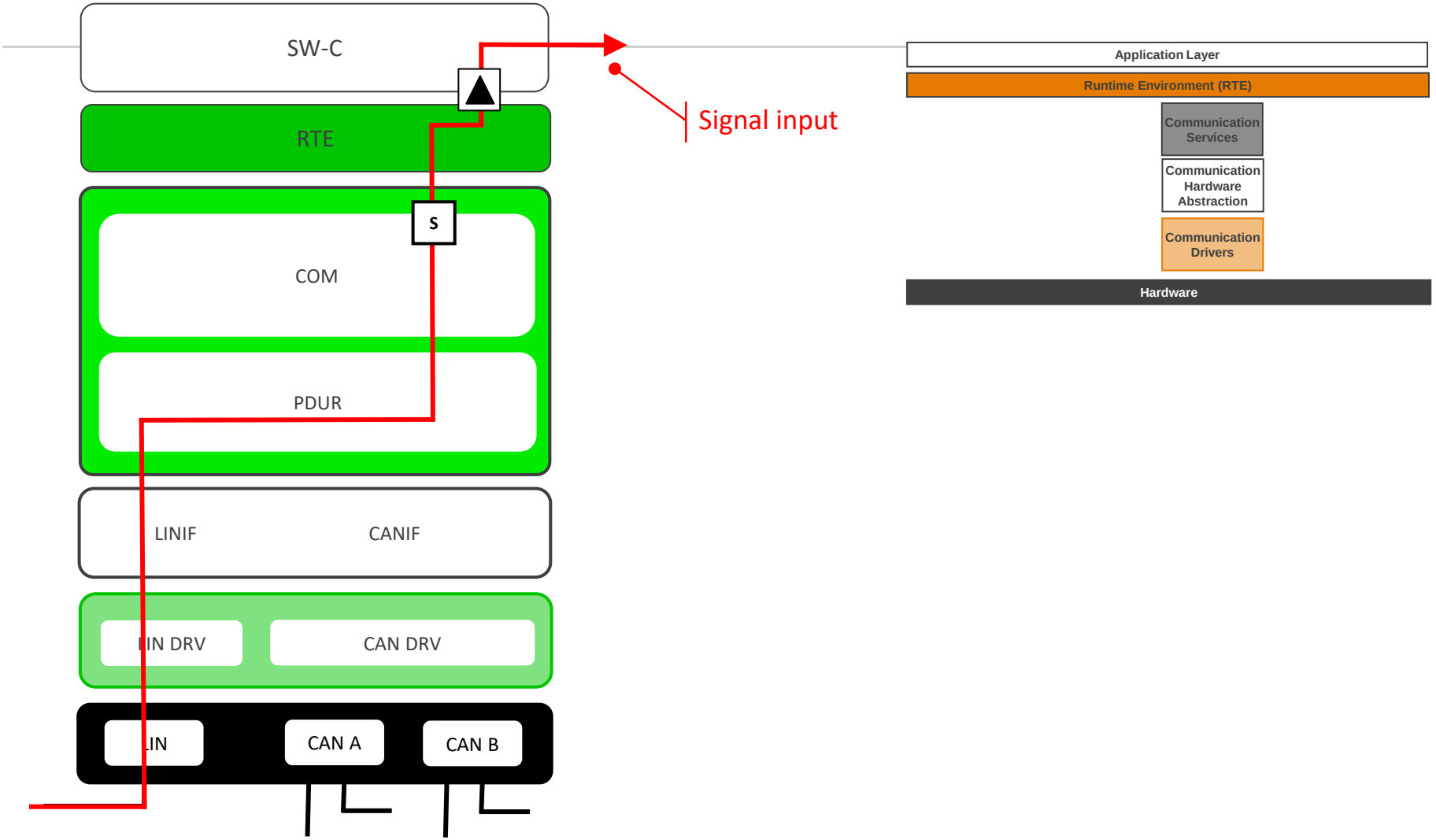
- COM
  - Abstract the signals from the PDU message
  - Provide a signal interface to the application layer
  - Provide signal gateway functionality

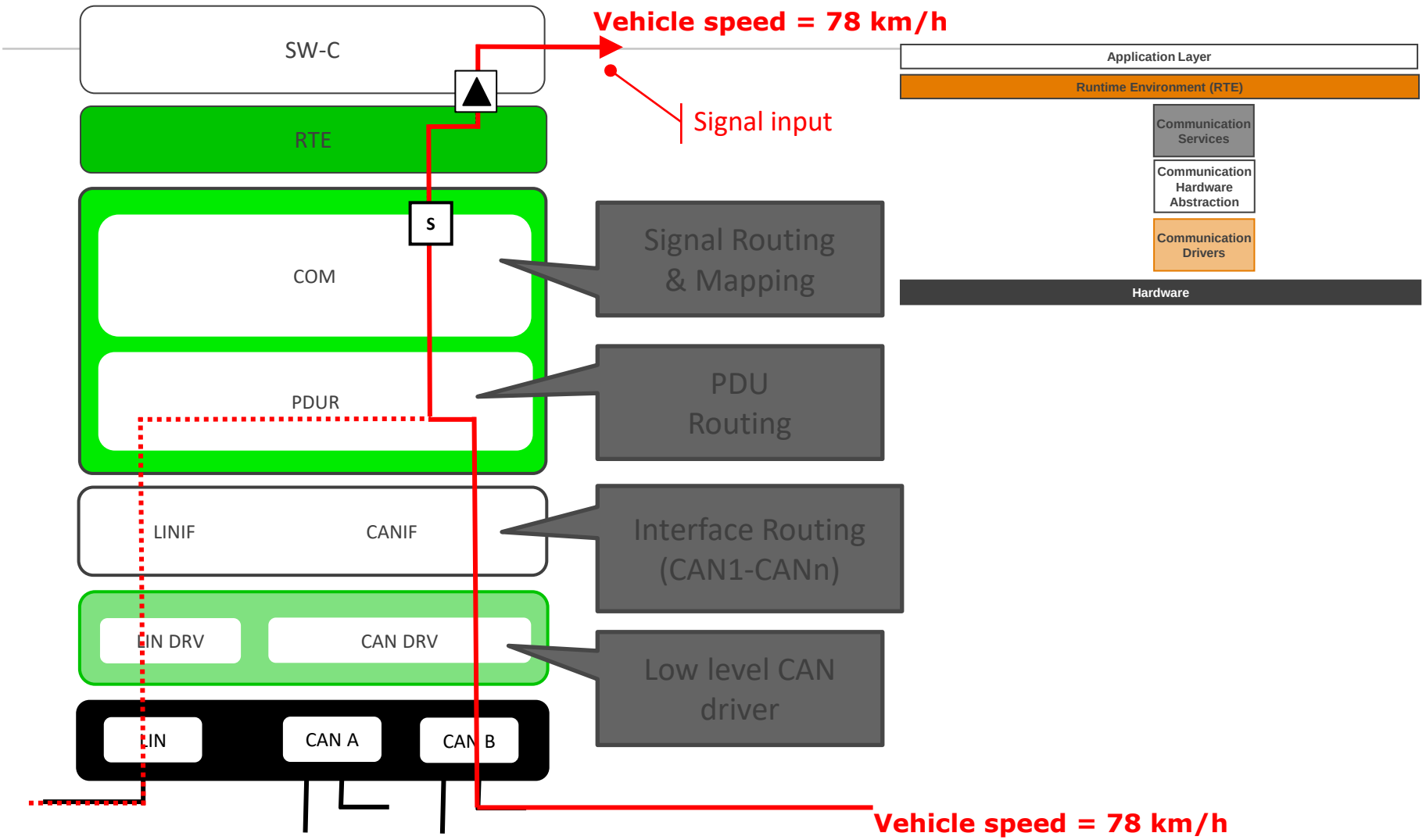
- PDUR
  - Abstract from the communication bus type (CAN, LIN)
  - Provide PDU message gateway functionality, both between the same and different kind of bus types
  - Provide multicasting functionality

- CANIF and LINIF
  - Abstract from the hardware driver(s)
  - Implements hardware independent but bus-specific functions
  - Handling message queuing

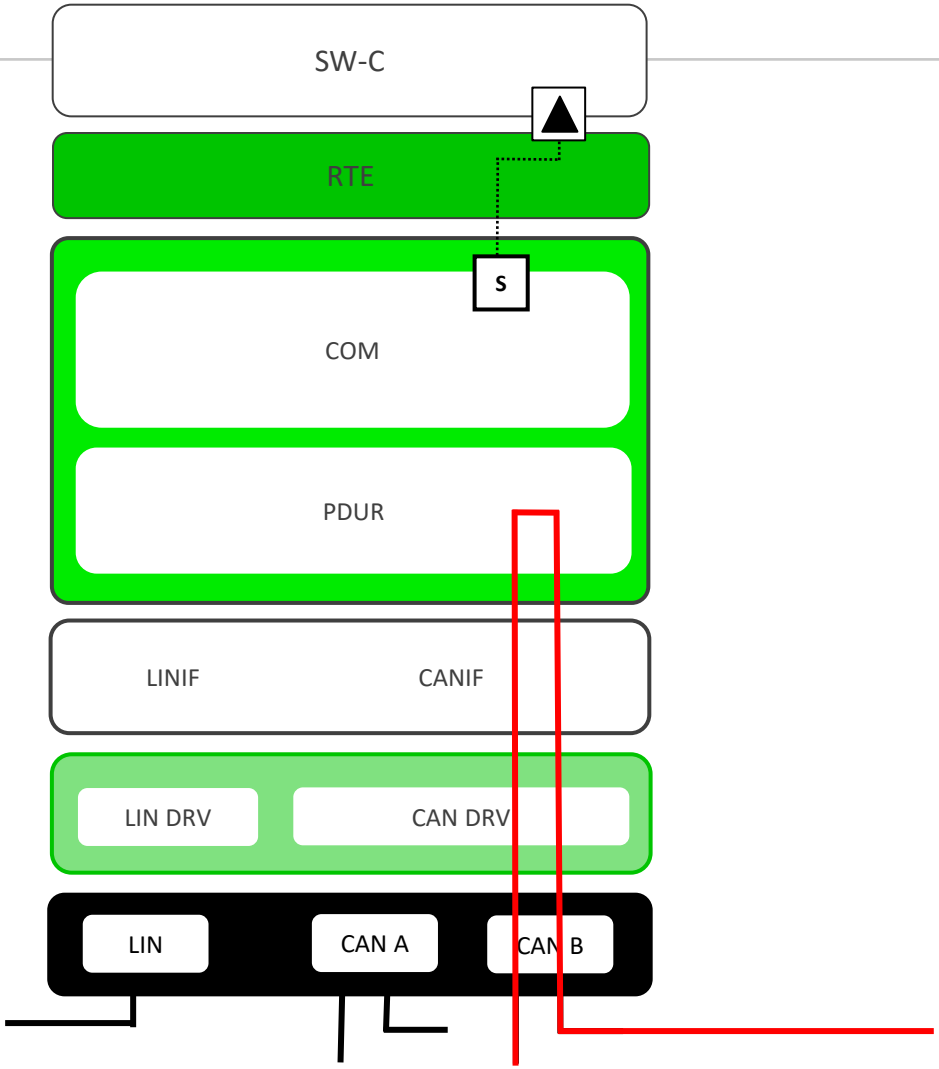
- IPDUM
  - Makes it possible to use different IPDU data layout for the same IPDU ID
  - This feature is already (before AUTOSAR) known from CAN communication

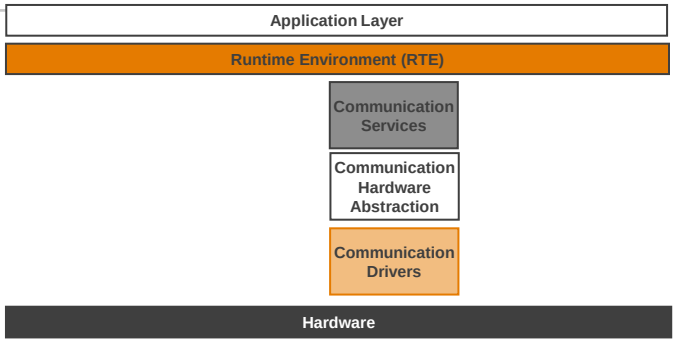
Architecture - Example 2: Communication (CAN / LIN)





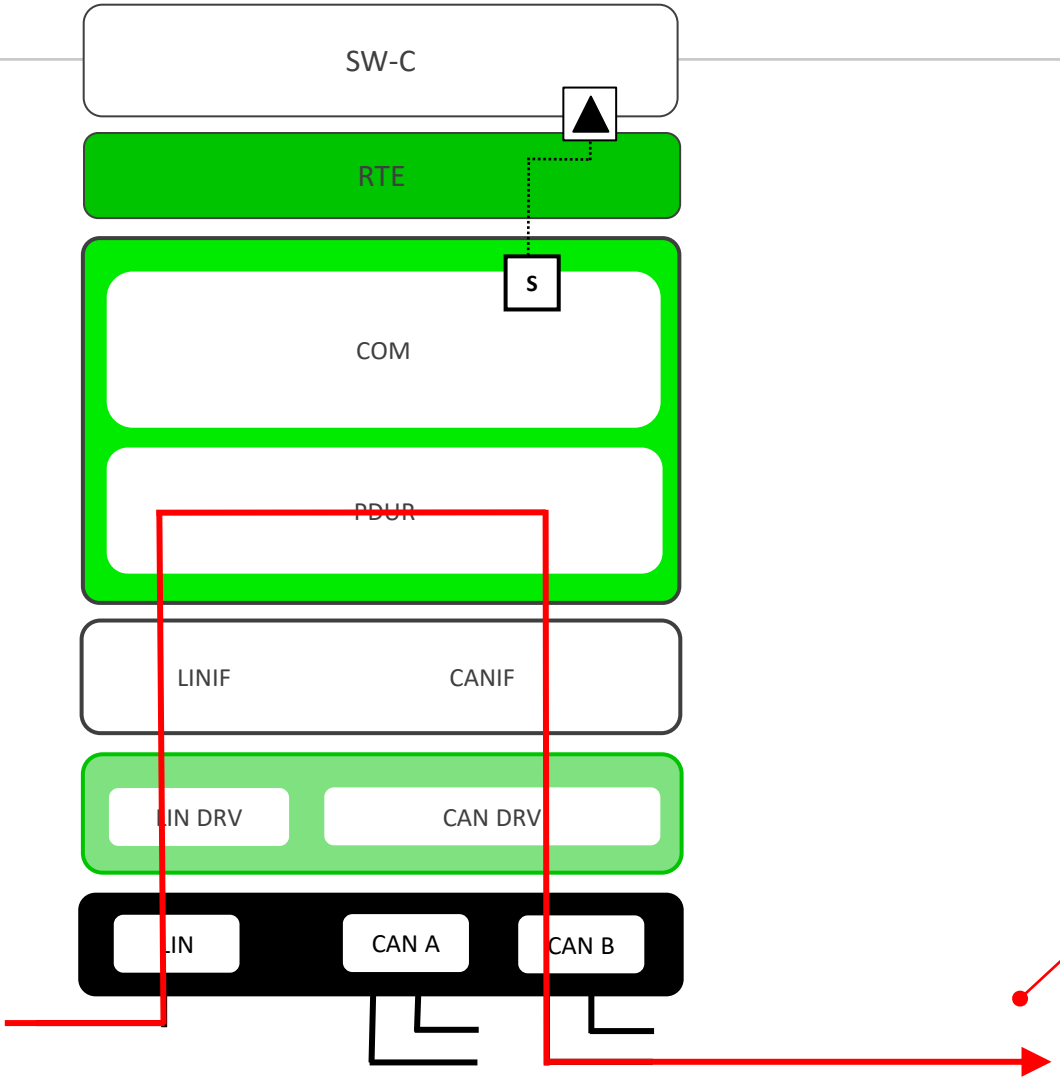
Architecture - Example 2: Communication (CAN / LIN)

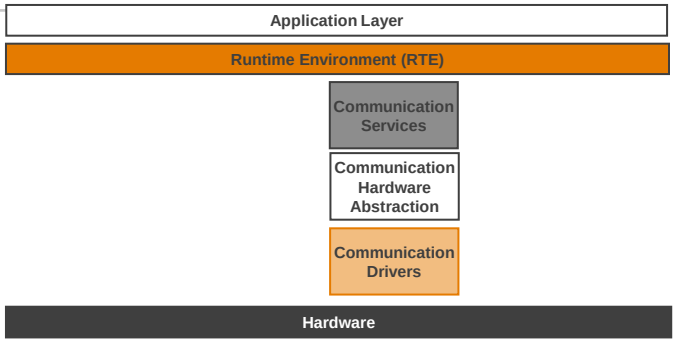




Message Gateway (CAN to CAN)

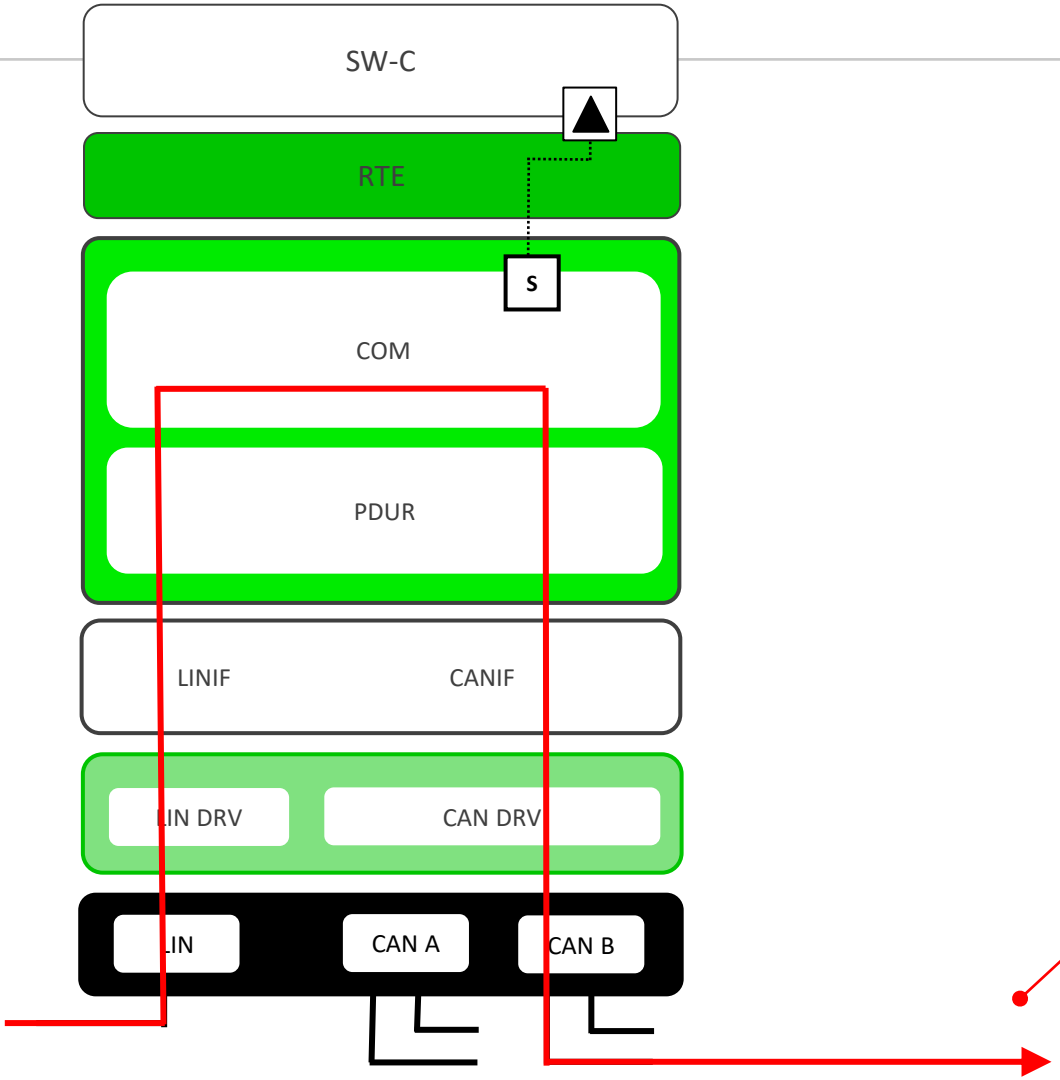
Architecture - Example 2: Communication (CAN / LIN)

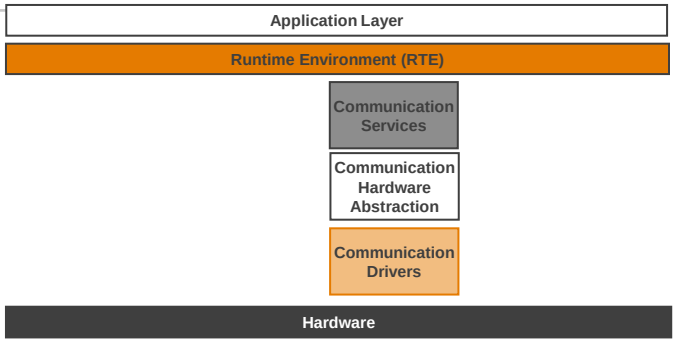




Message Gateway (LIN to CAN)

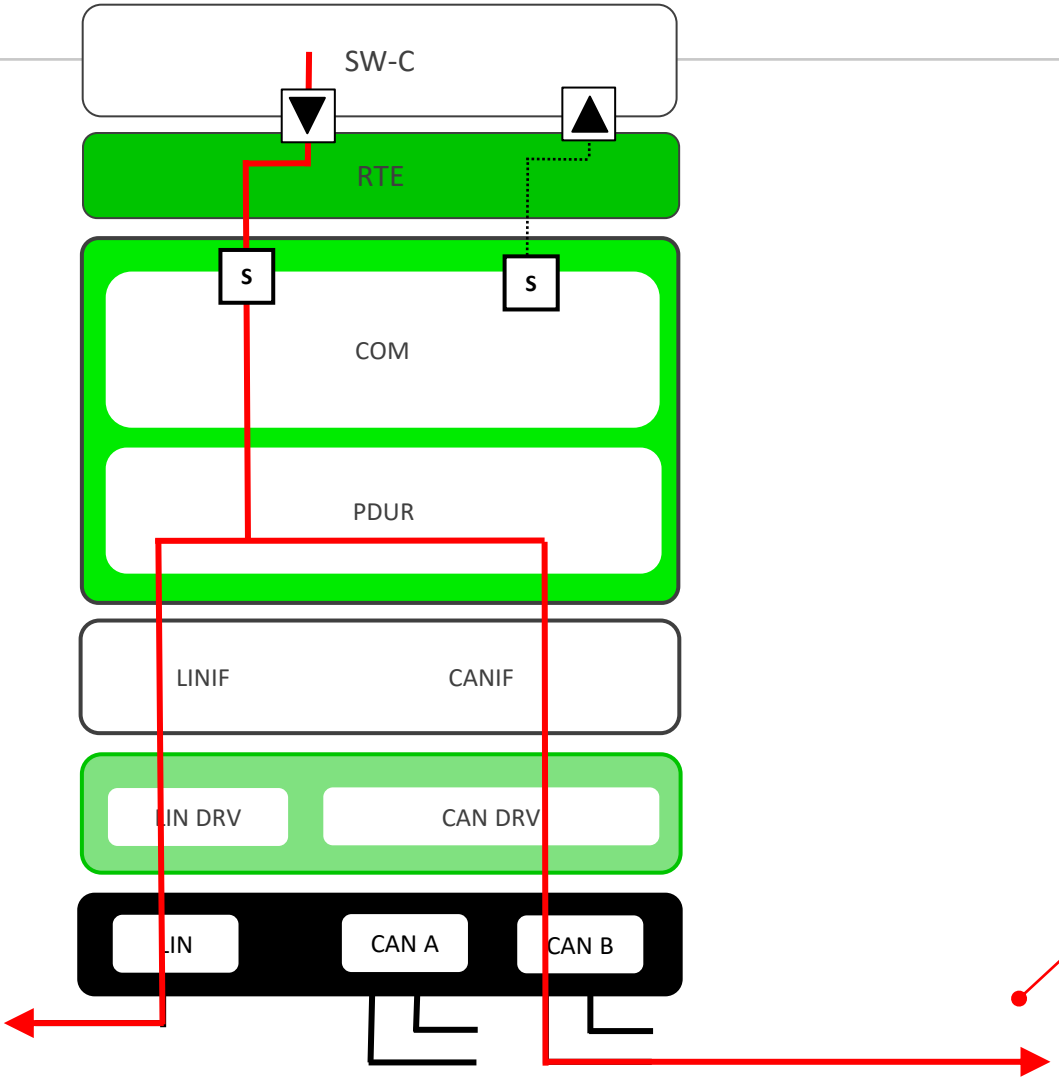
Architecture - Example 2: Communication (CAN / LIN)

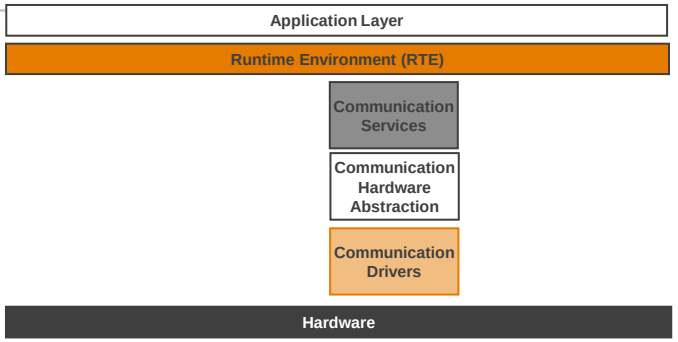




Signal Gateway (LIN to CAN)

Architecture - Example 2: Communication (CAN / LIN)



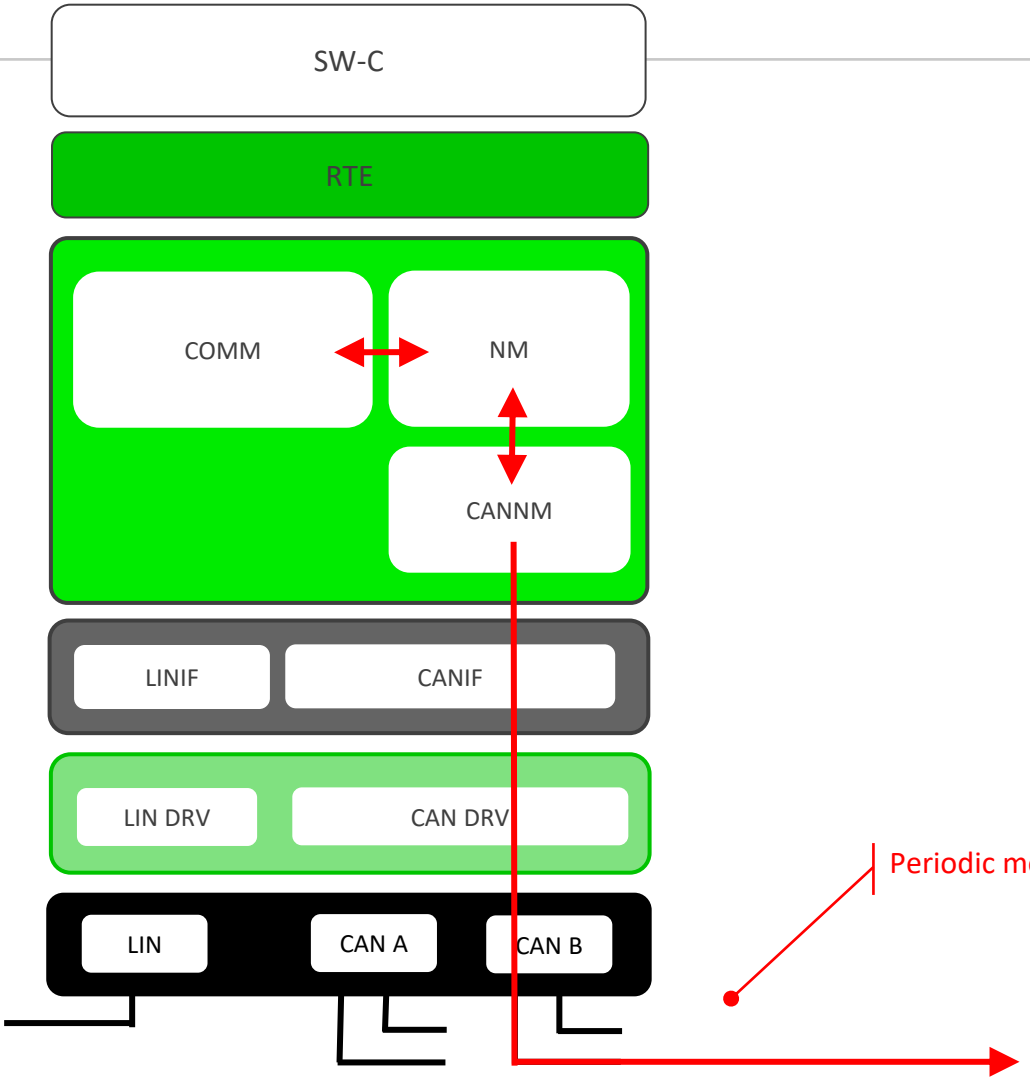


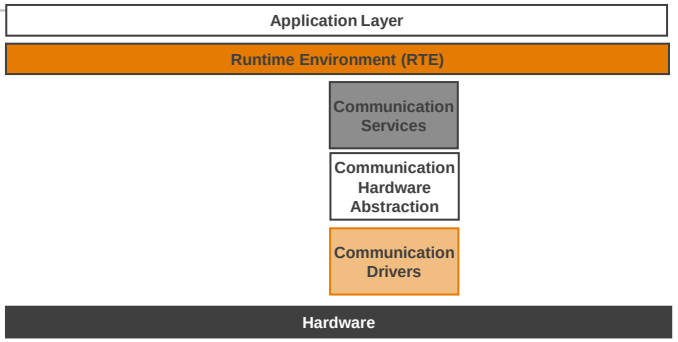
Multicast (LIN and CAN)

- NM
  - Abstract from bus specific network management
  - Coordinate busses (synchronously shut down)
  - Provide interface to ComM (ComM does *not* interface to CANNM)
  - Support for NM coordinator functionality (optional)

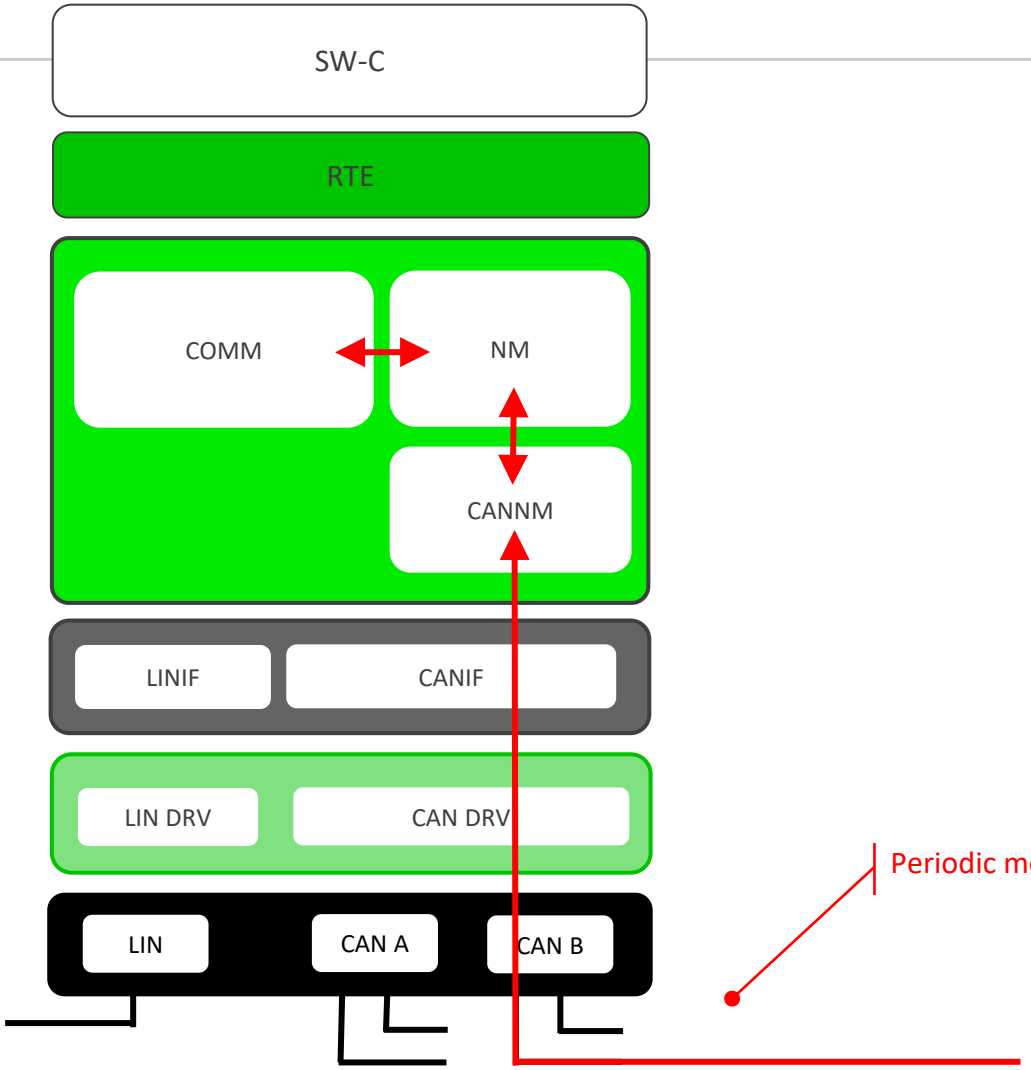
- CANNM
  - CANNM is a decentralized network management (no master / slave)
  - Each node has a dedicated NM message
  - The NM message is sent periodically and as long as the CAN bus is requested (needed)
  - If no NM messages are received for a (configurable) amount of time bus sleep shall be entered
  - A transmission mode with bus load reduction is available (optional)
  - A passive, “listen only”, mode is available (configurable)

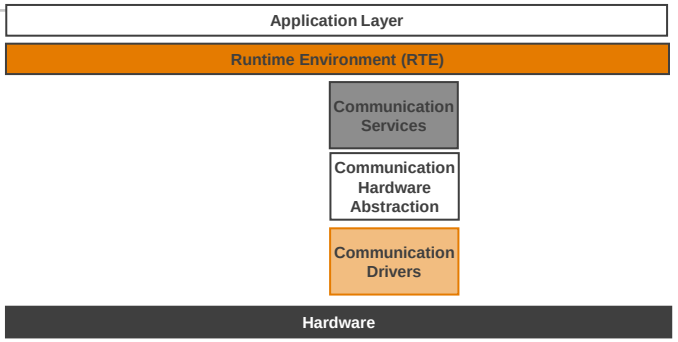
- CANIF
  - Abstract from the hardware driver(s)
  - Implements hardware independent but bus-specific functions
  - Handling message queuing





Architecture - Example 3: Network Management (CAN)



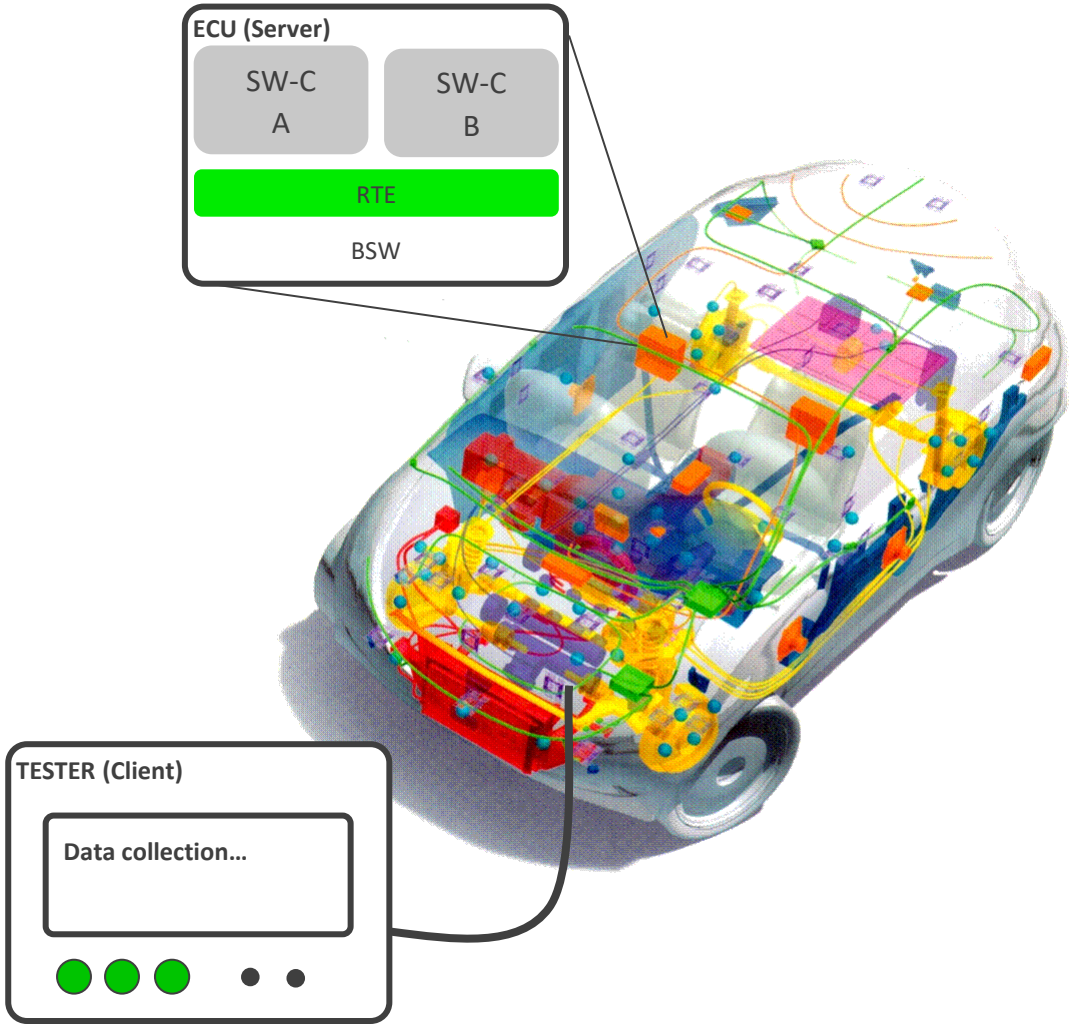


EB tresos classic AUTOSAR training - Diagnostic

Elektrobit

Introduction to Diagnostics

- What is diagnostics
  - Fault management
  - Calibration
  - Programming
  - Function check
  - Authorities demand (OBD emission related)
- Server (ECU)
  - Service provider
  - Collects data
  - Executes services
- Client (Tester)
  - Asks for services
  - Presents information



Chapter overview

- Default (Development) Error Tracer (DET)
- Diagnostic Event Manager (DEM)
- Function Inhibition Manager (FIM)
- Diagnostic Communication Manager (DCM)

Det Functionality

- Additional **checks** in the BSW (to detect wrong parameter value, buffer overflow, wrong sequence of calls, etc.)
- Enabled in configuration of **each BSW module**
- Also usable for **application**
- All default (development) errors are reported to Det:

```
Det_ReportError( ModuleId,
InstanceId, ApiId, ErrorId )
```

DEM Functionality

- Event **status** and data processing and storage
- Diagnostic protocol interface (to DCM)
- **Operation cycle** management (for event qualifying)
- Event **aging**
- Status/Data-change & further **callbacks**
- Freeze Frame (data) **prestorage**
- **Enable condition** handling (for event qualifying)
- Event report **debouncing** (for pre-passed / pre-failed)
- **Indicator** processing
- **OBD**-related functionality

FiM Functionality

- Calculates and summarizes **permission/inhibition conditions** on defined functionalities (FIDs)
- Each permission/inhibition condition is based on the **status of a diagnostic event** (from Dem)
- **Retrieves event status** by Dem_GetEventStatus() *or* is triggered by the Dem on event status change
- **Provides permission result** (of FID) to SW-Cs and BSW modules
- `FiM_GetFunctionPermission( FID, *Permission )`

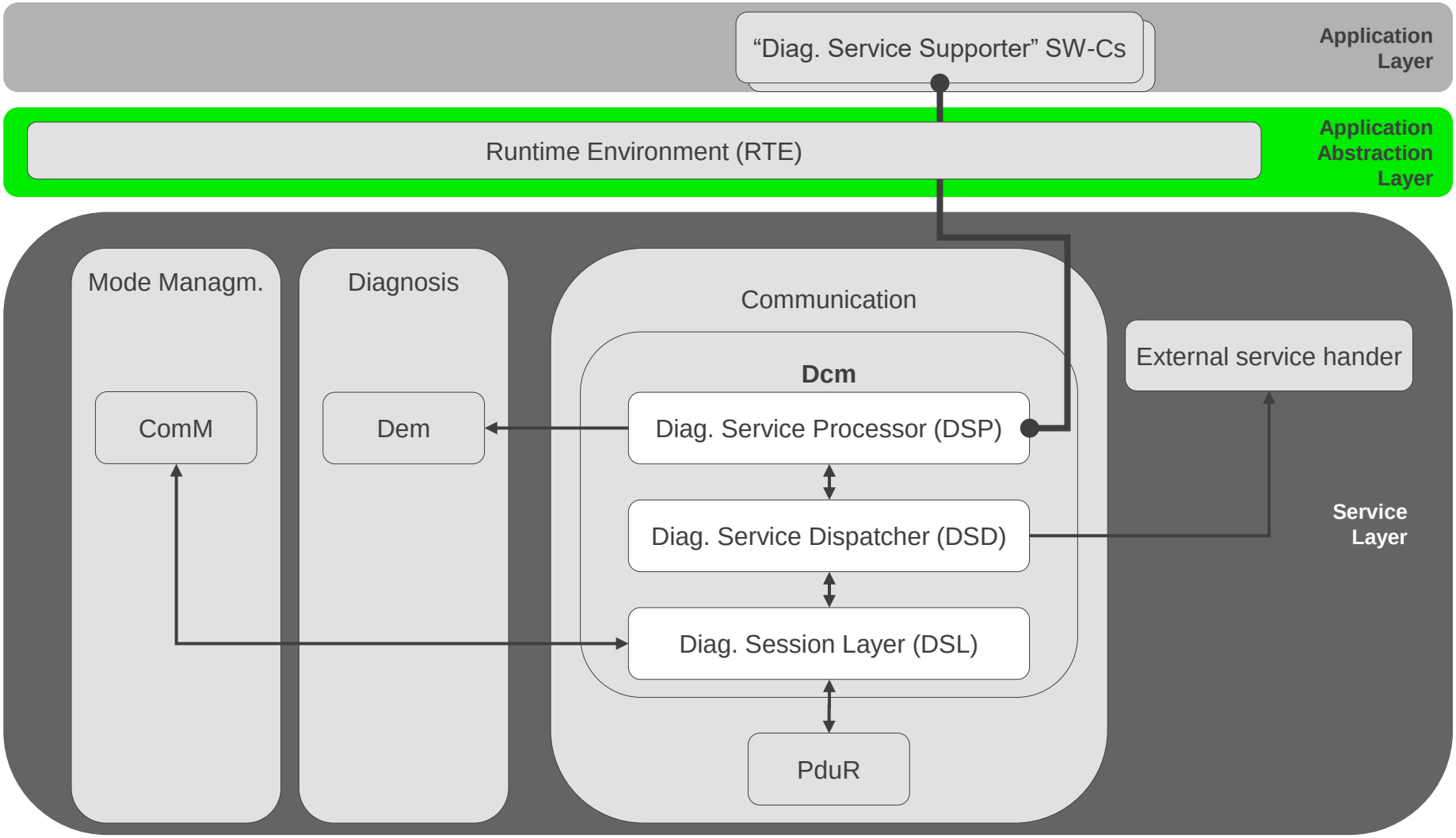
DCM Functionality

- Acts as mediator between diagnostic services and the vehicle network communication (CAN, LIN, Flex Ray)

- Receives/Sends diagnostic messages from/to the PDU Router (PduR)

Implements basic support of the **UDS protocol** (e.g. pos./neg. response handling, physical and functional request handling, …)

- Provides (or triggers) the appropriate service processors which retrieve required information from the application



EB tresos classic AUTOSAR training - Communication stack

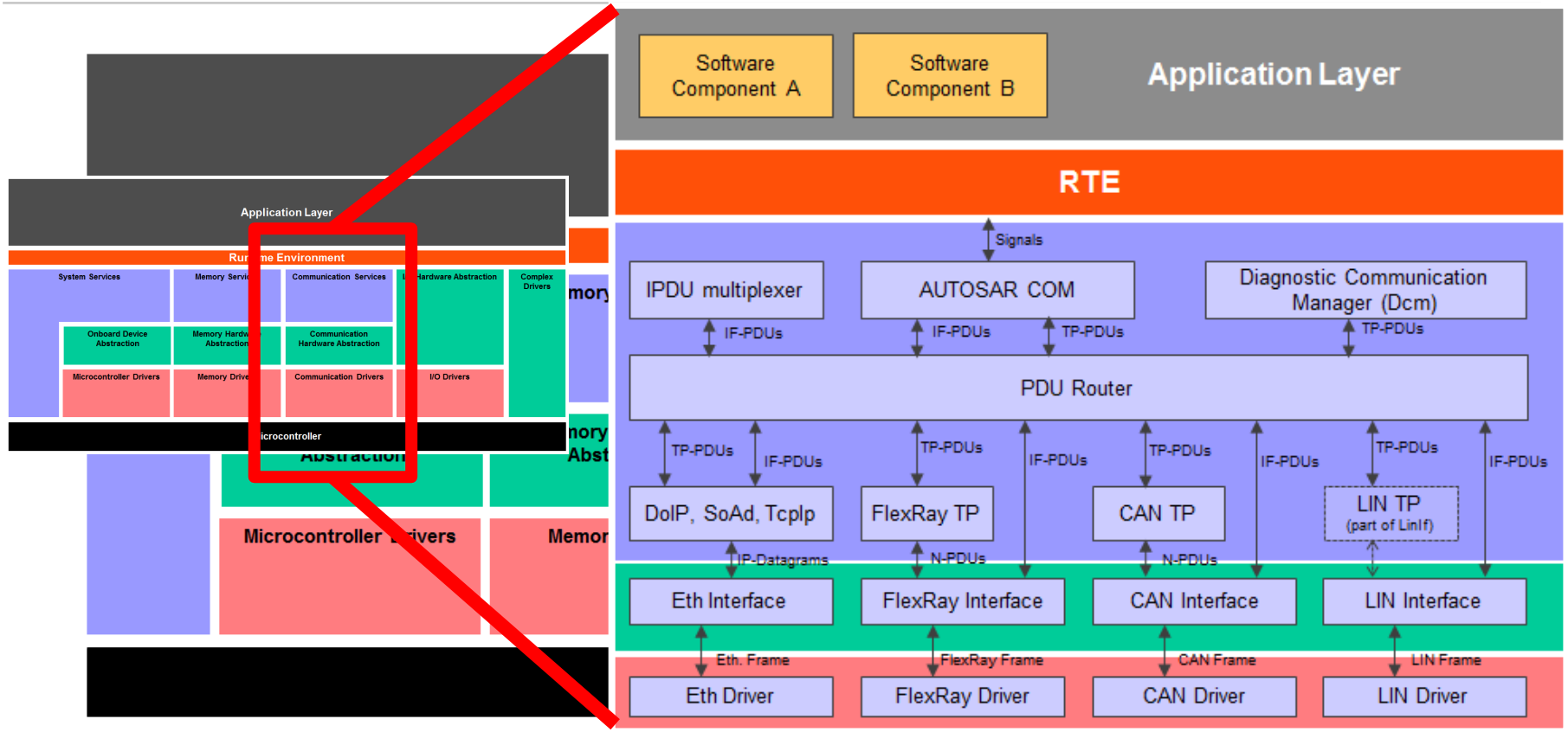
Elektrobit

Chapter overview

- Com
- PduR, IpduM
- Can Stack
- FlexRay Stack
- Ethernet/IP Stack

AUTOSAR Communication Stack
Architecture Overview (1/2)

Elektrobit



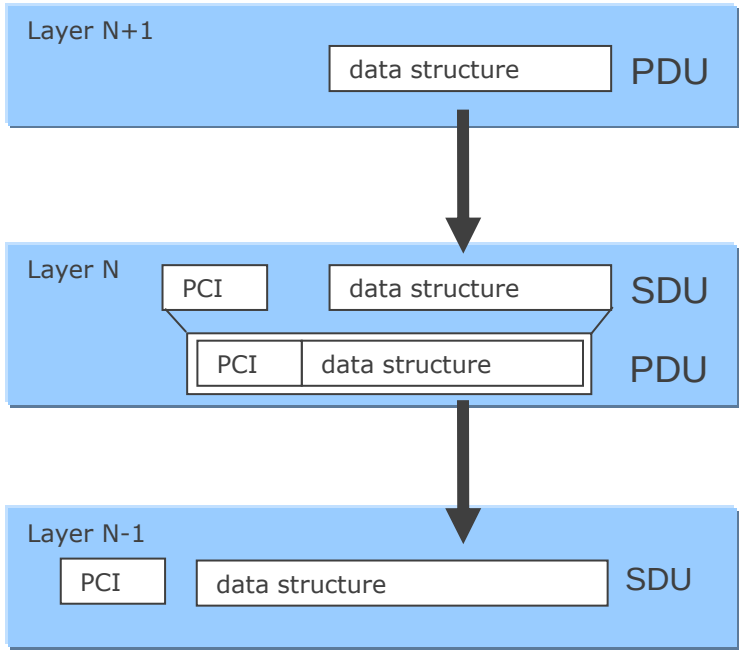
Com Stack Terminology

- **Signal**:Basic communication object
  primitive data types (int, char, etc) supported
  - Signal Group:
    complex data types (struct) supported

- **PDU**: Protocol Data Unit
  Basic unit for data transfer. Signals are bundled into PDUs for transmission
  - Prefix I-, N-, L-

Data packing

- **SDU**
  Service Data Unit
- **PCI**
  Protocol Control Information
- **PDU**
  Protocol Data Unit



Signal flow – non diagnostics

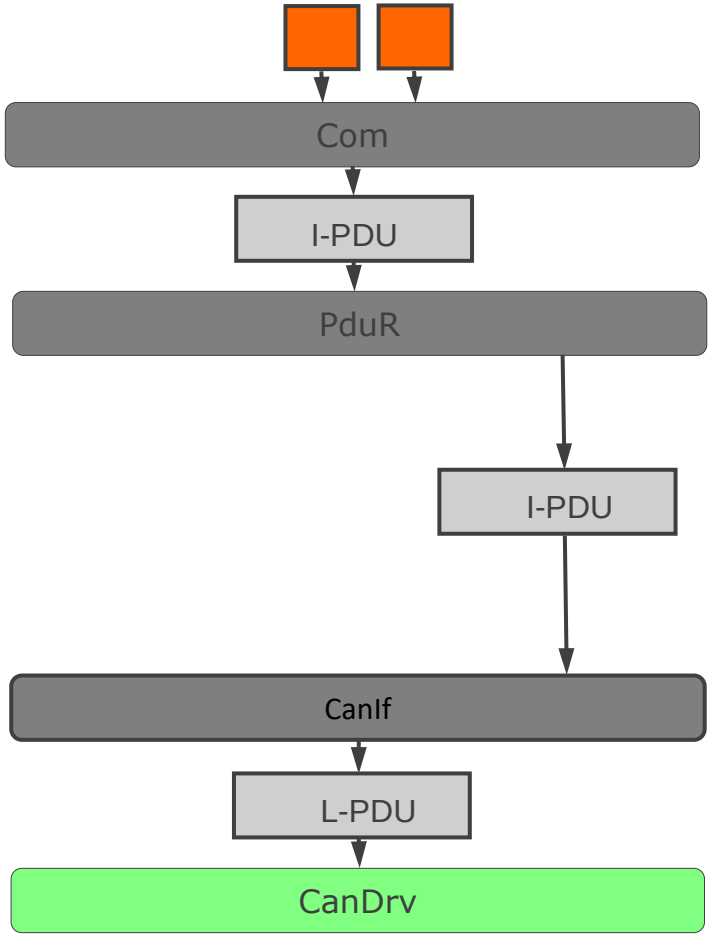
Signals

Interaction layer PDU (packed signal values)

Interaction layer PDU

Data link layer PDU

Associated hardware object handle (HOH)



Signal flow - diagnostics

Interaction layer PDU

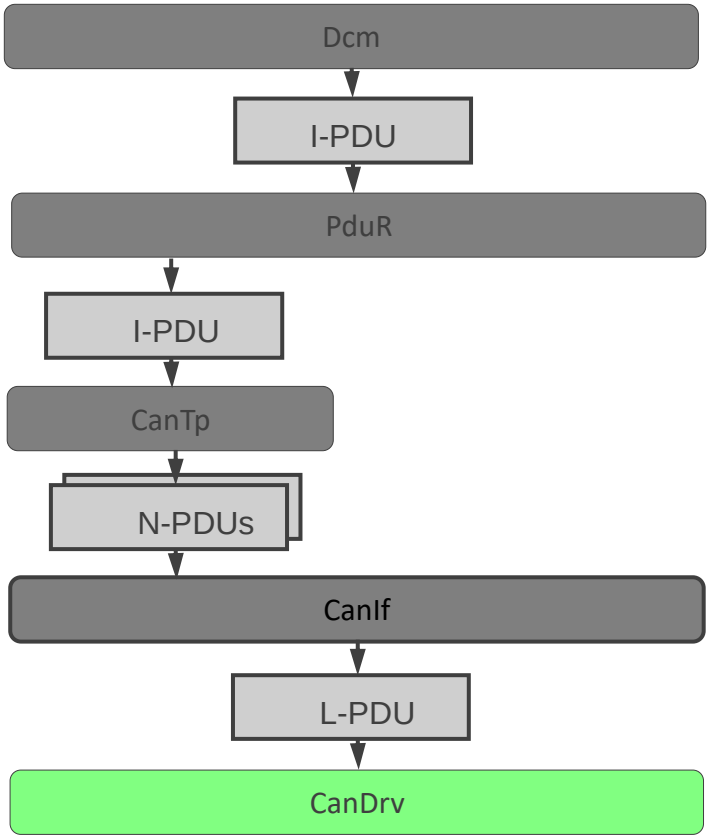
Interaction layer PDU

Network layer PDUs (segmented I-PDU)

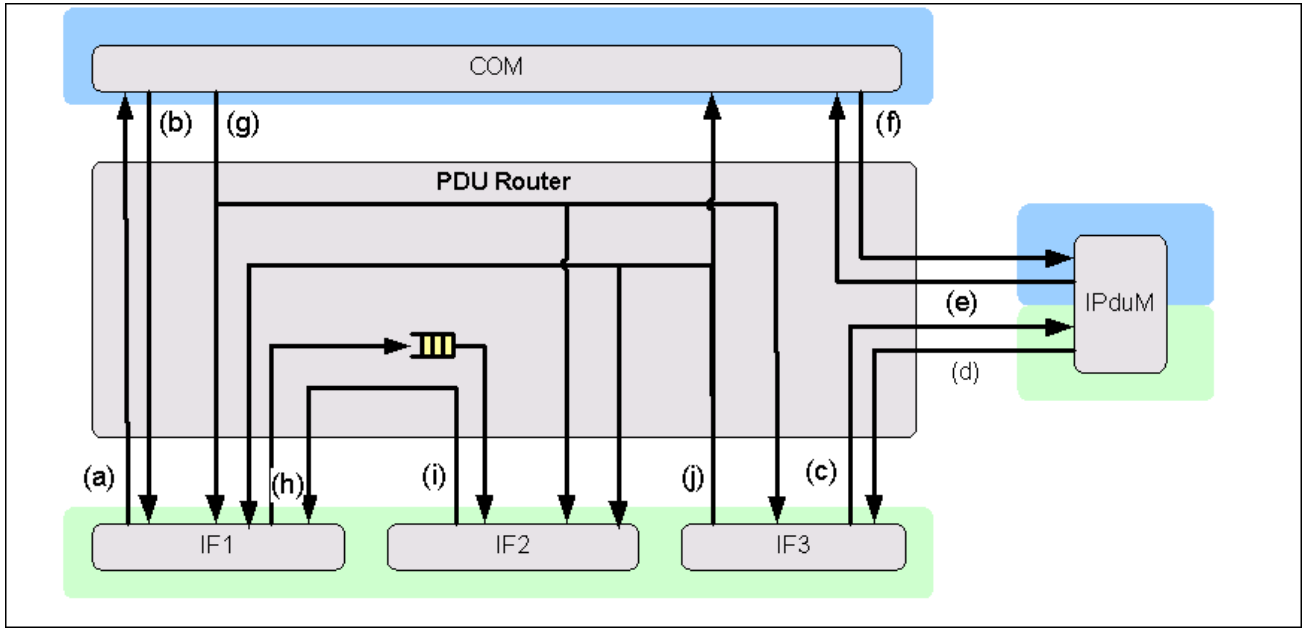
PCI added to data fields

Data link layer PDU

Associated hardware object handle (HOH)

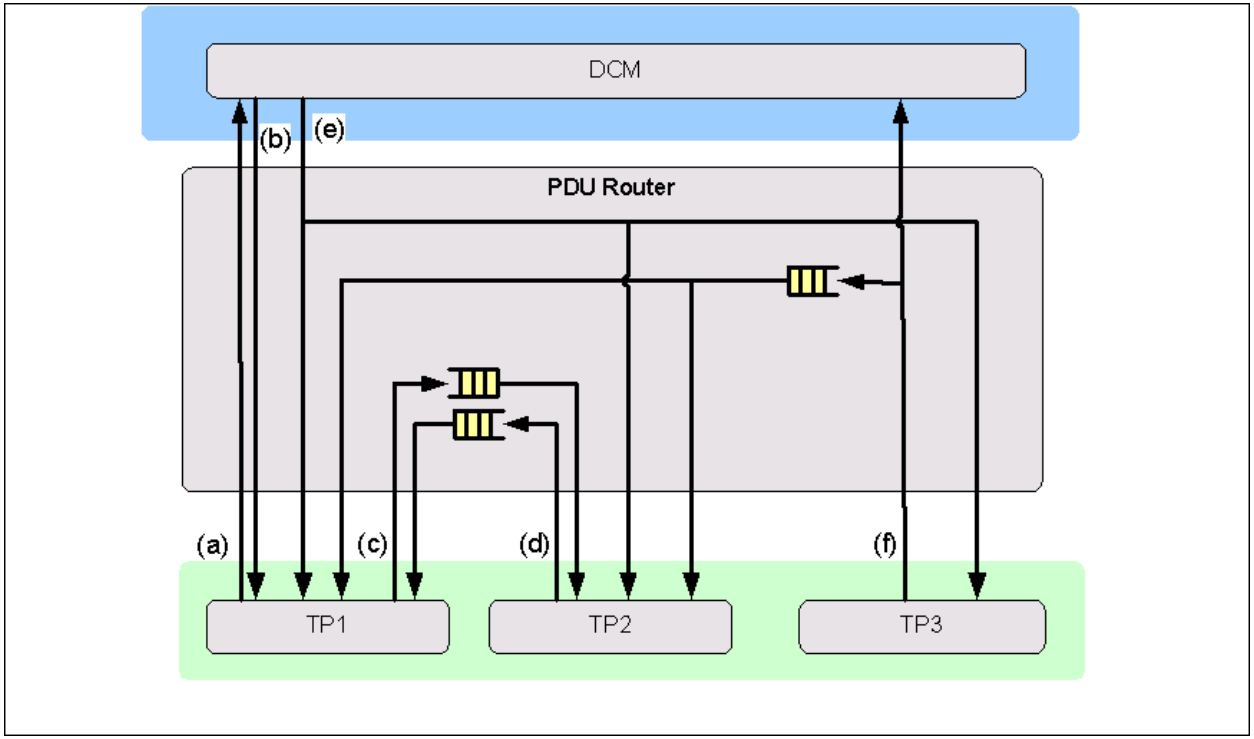


PduR – IF Routing operations



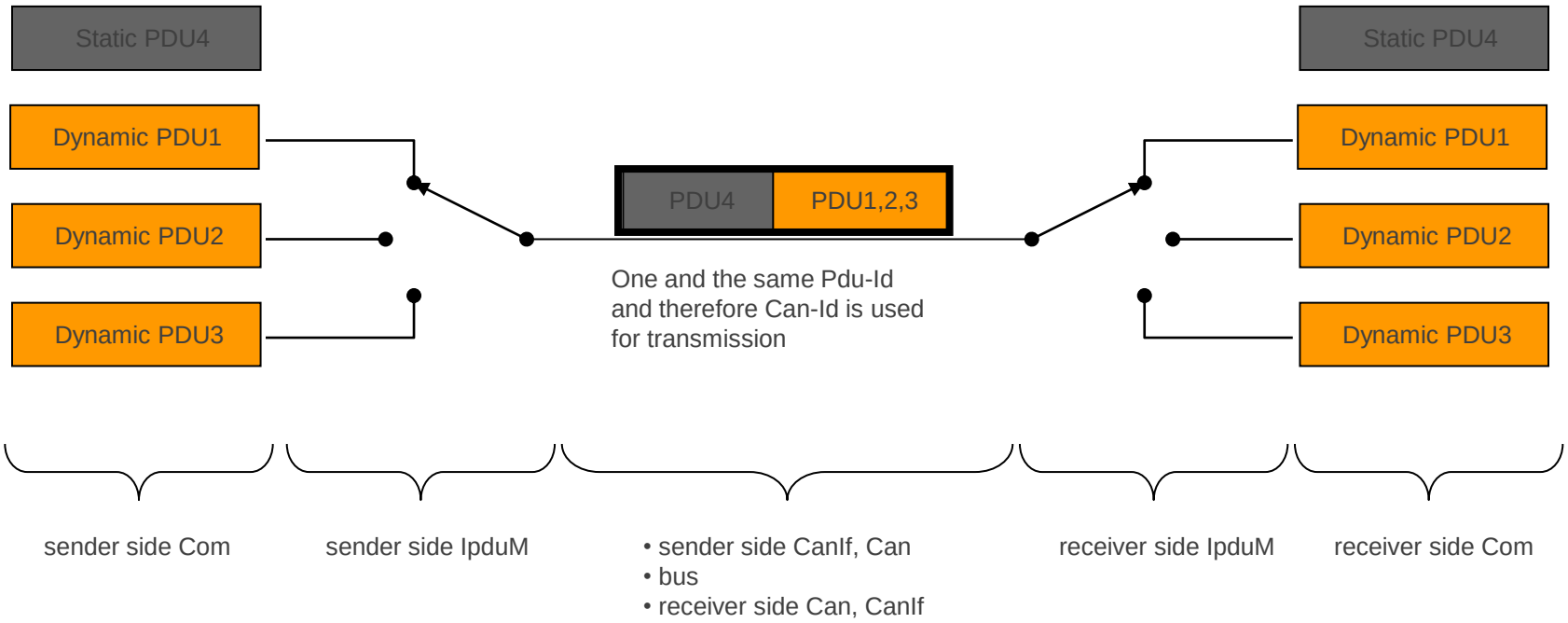
- Standard Routing Operations (a), (b), (c), (d), (e) and (f)
- Multicast Routing Operation (g)
- Gateway Routing Operations ((h), (i) as well as the multicast gateway routing operation (j)

PduR – TP Routing operations

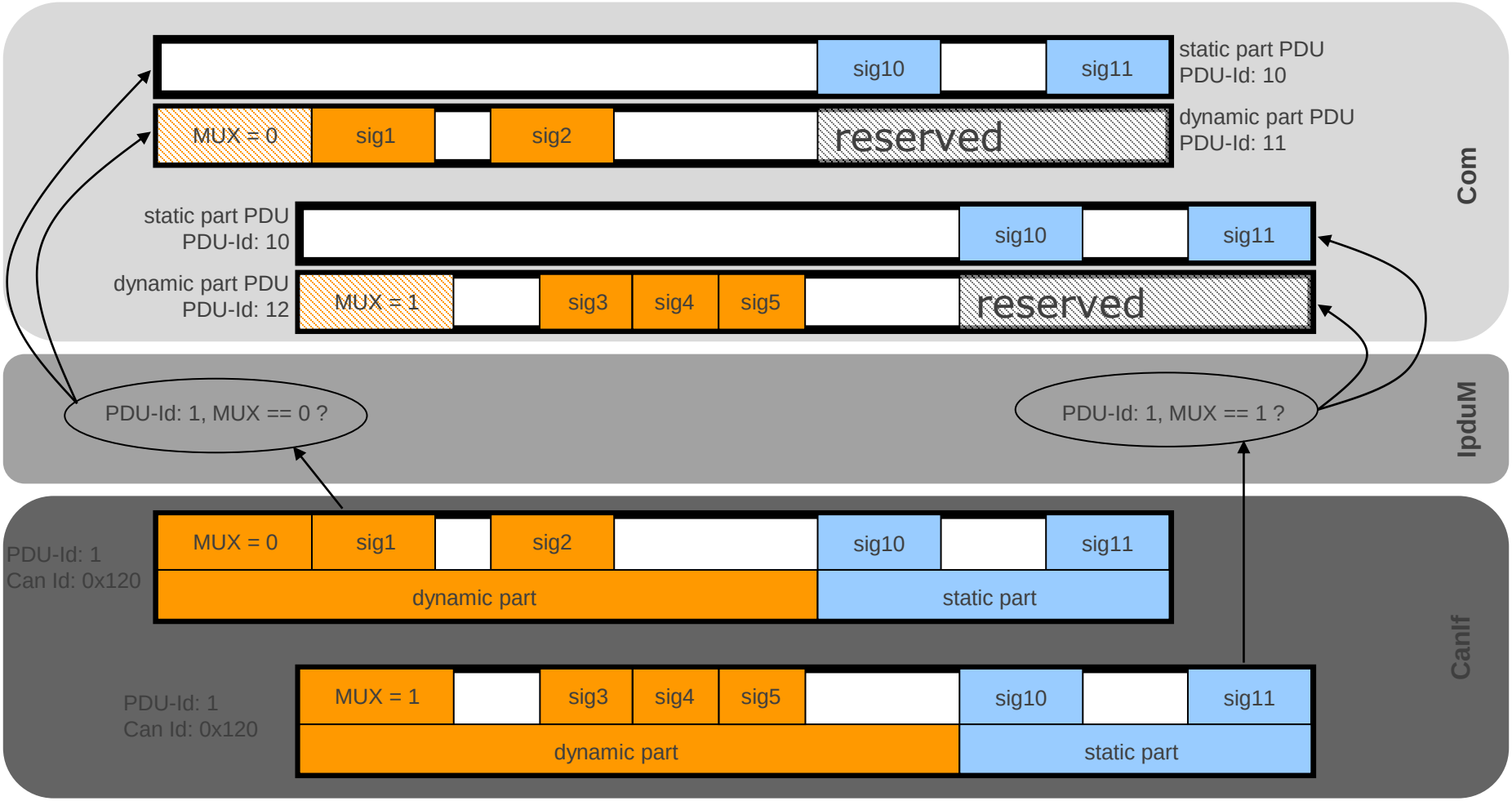


- Standard Routing Operations (a), (b)
- Multicast Routing Operation (e)
- Gateway Routing Operations ((c), (d) and multicast gateway routing operation (f)

IpduM principle



IpduM – I-PDU layout



CAN STACK Functionality

- Mapping PDUs to HOH

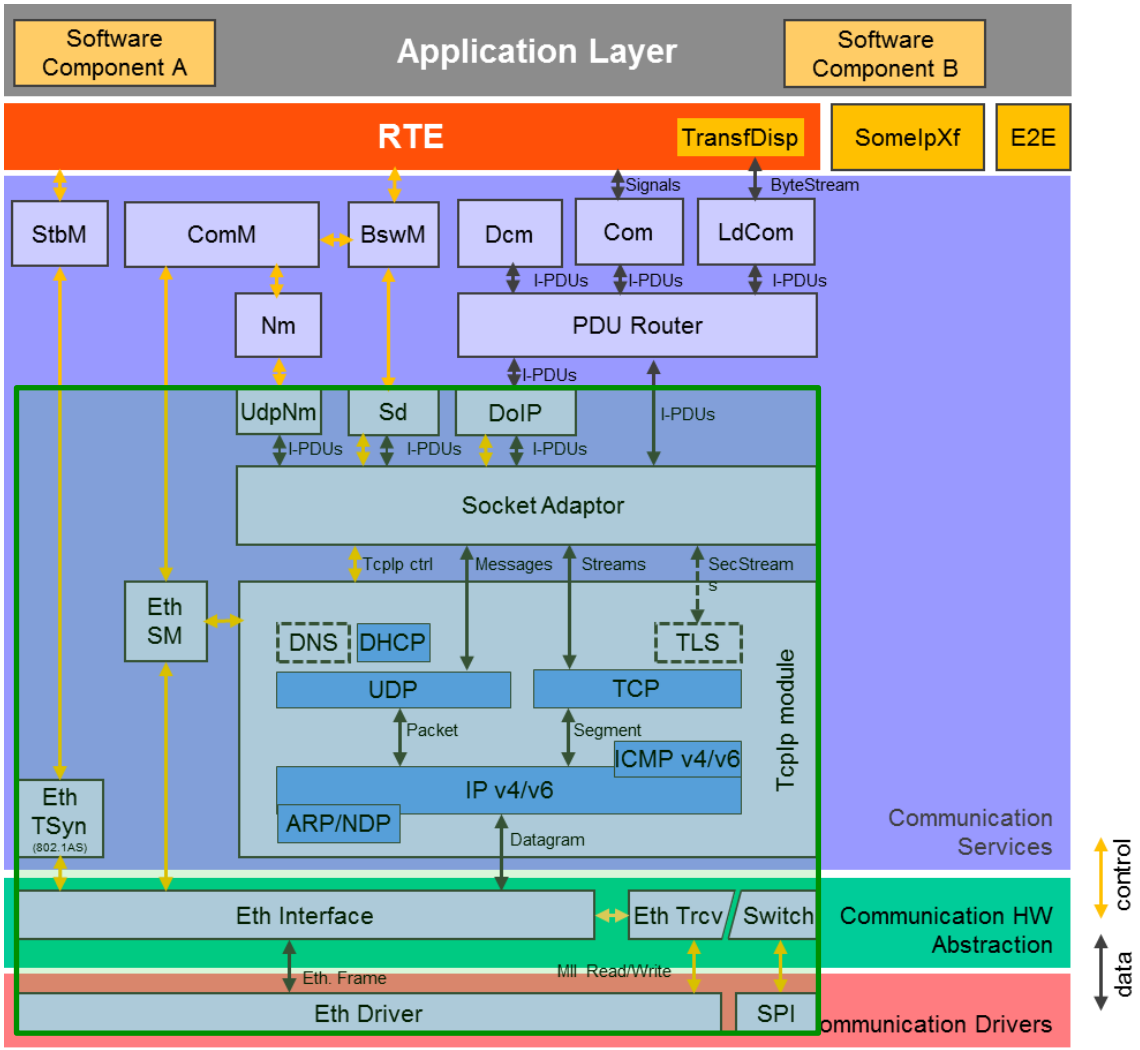
Hardware abstraction by hardware object handles (HOH)

CanIf maps PDUs onto hardware object handles

One HOH represents one message buffer of a Can controlle

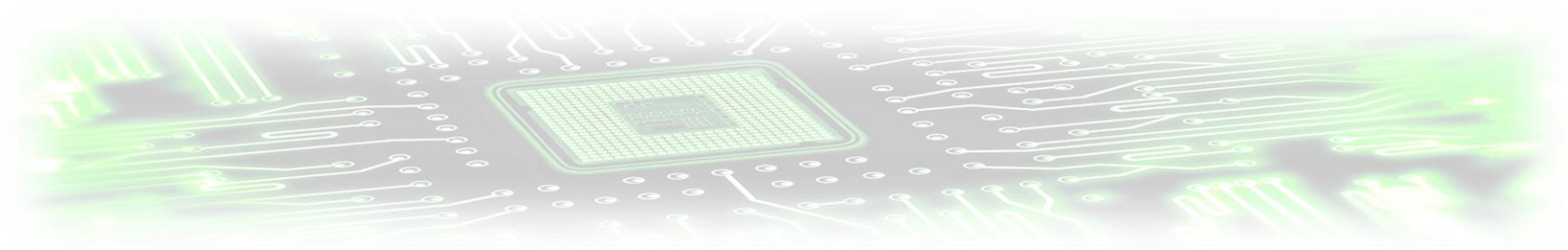
- Transmit buffering/ Cancellation
- Interrupt / Polling mode
- CanTP provides services for
  - Segmentation of data in transmit direction
  - Collection of data in receive direction
  - Provides 1:1 and 1:n communication
  - Control of data flow
  - Detection of errors in segmentation sessions
- CAN/CANFD supported

AUTOSAR Ethernet Stack Overview



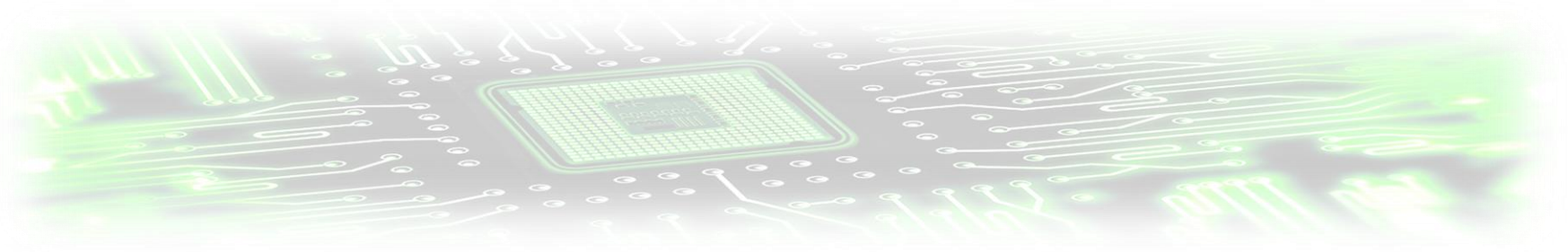
Transformers - Key Idea

- Pluggable transformer modules invoked by **`Rte`**
  - Carrying out **arbitrary data transformations**
    - E.g., safety transformer ➔ computation/verification of end-to-end checksum (**`E2EXf`**)
    - E.g., security transformer ➔ computation/verification of message authentication code
    - E.g., serializing transformers ➔ conversion of a complex data element into a byte stream
- Sequence of transformer modules defining a **processing chain** (i.e., transformer chain)
  - Output of one transformer serves as input to the next (similar to UNIX pipes)
  - **Serializing transformer** is always the **first/last** transformer in a chain
- Invoked in the context of the **`Rte_Read/Write()`** API

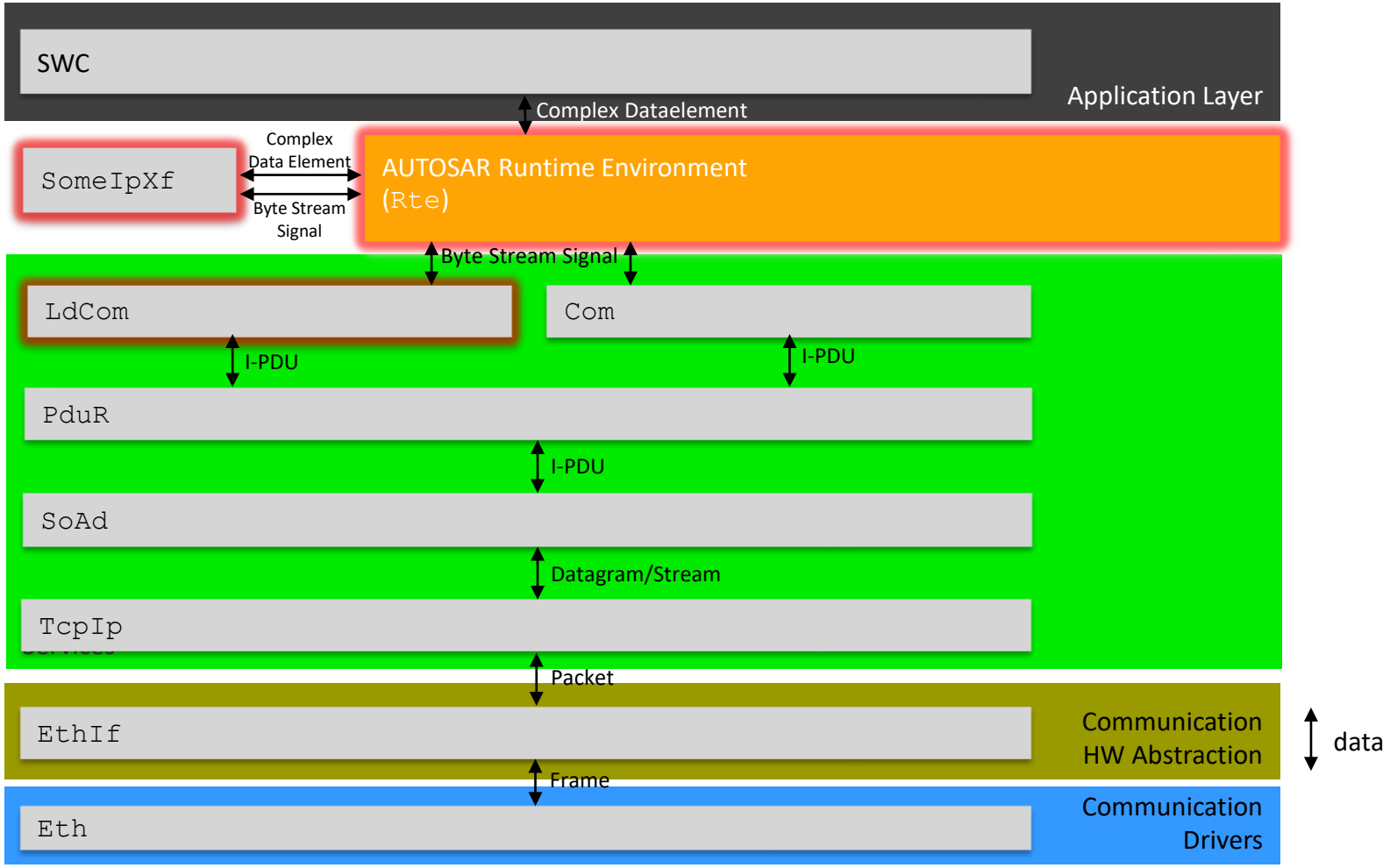


Serializing Transformers

- Conversion of a (complex) data elements into a byte stream
  - **Flattening of structured data types**
  - Endianess conversions (configurable – big endian is default)
  - Support for **variable length arrays**
- Two flavors of serializing transformers in AUTOSAR
  - **`SomeIpXf`**
  - **`ComXf`** (limited to S/R communication like standard Com)

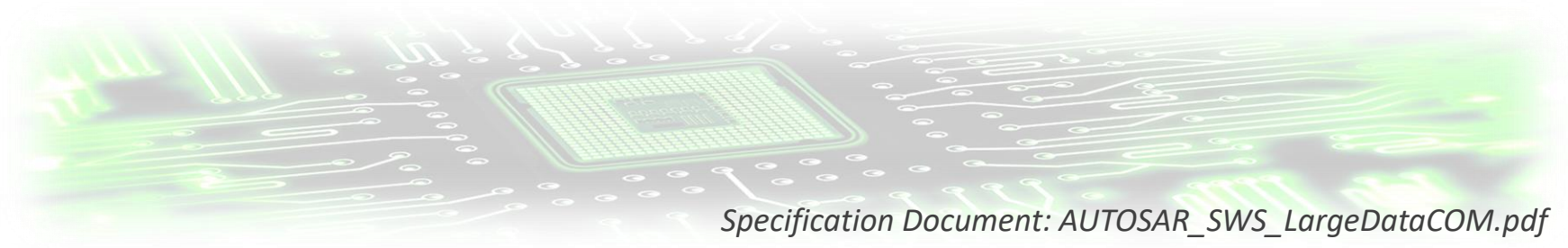


Architectural Overview - SomeIpXf



Large Data Com (`LdCom`)

- Formerly named „efficient Com (**`Ecom`**)“
- No need for signal-based packing if we already have a byte stream
- **Stripped-down `Com` module**
  - No filtering
  - No transmission modes
  - No signal packing
  - No internal buffering
- **IF and TP archetype interface towards RTE**
  - Changes communication paradigm from „pull“ (**`Com_ReceiveSignal()`**) to „push“ (**`Rte_LdComCbkRxIndication()`**)



Specification Document: AUTOSAR_SWS_LargeDataCOM.pdf

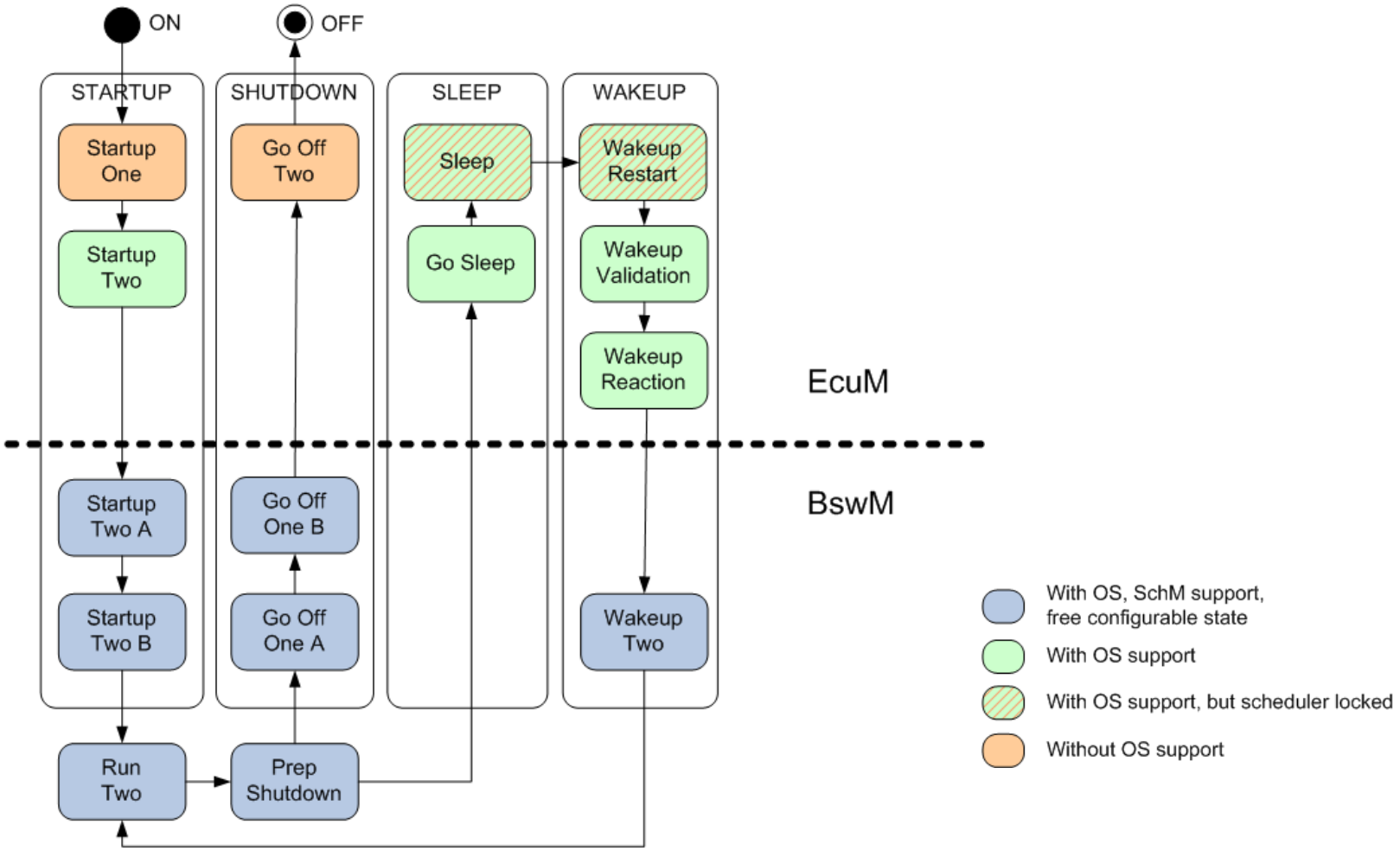
EB tresos classic AUTOSAR training - Mode management

Elektrobit

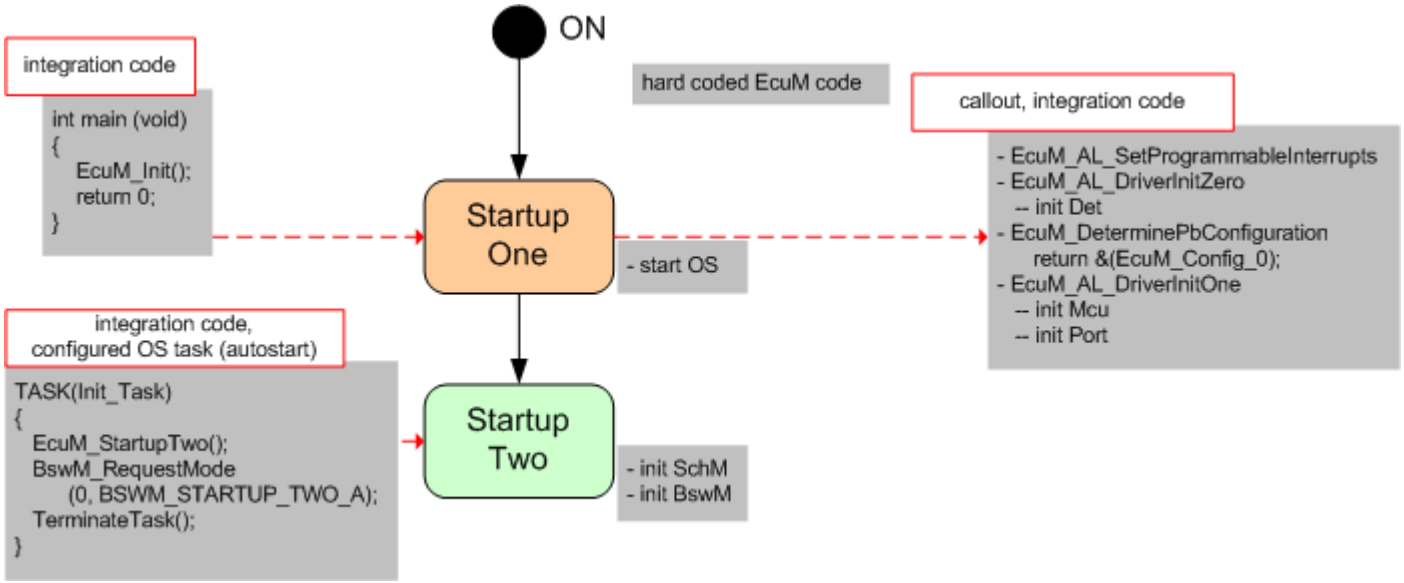
Chapter overview

- ECU State- BSW Mode- Manager
- Watchdog Management
- COM Manager
- Network Management

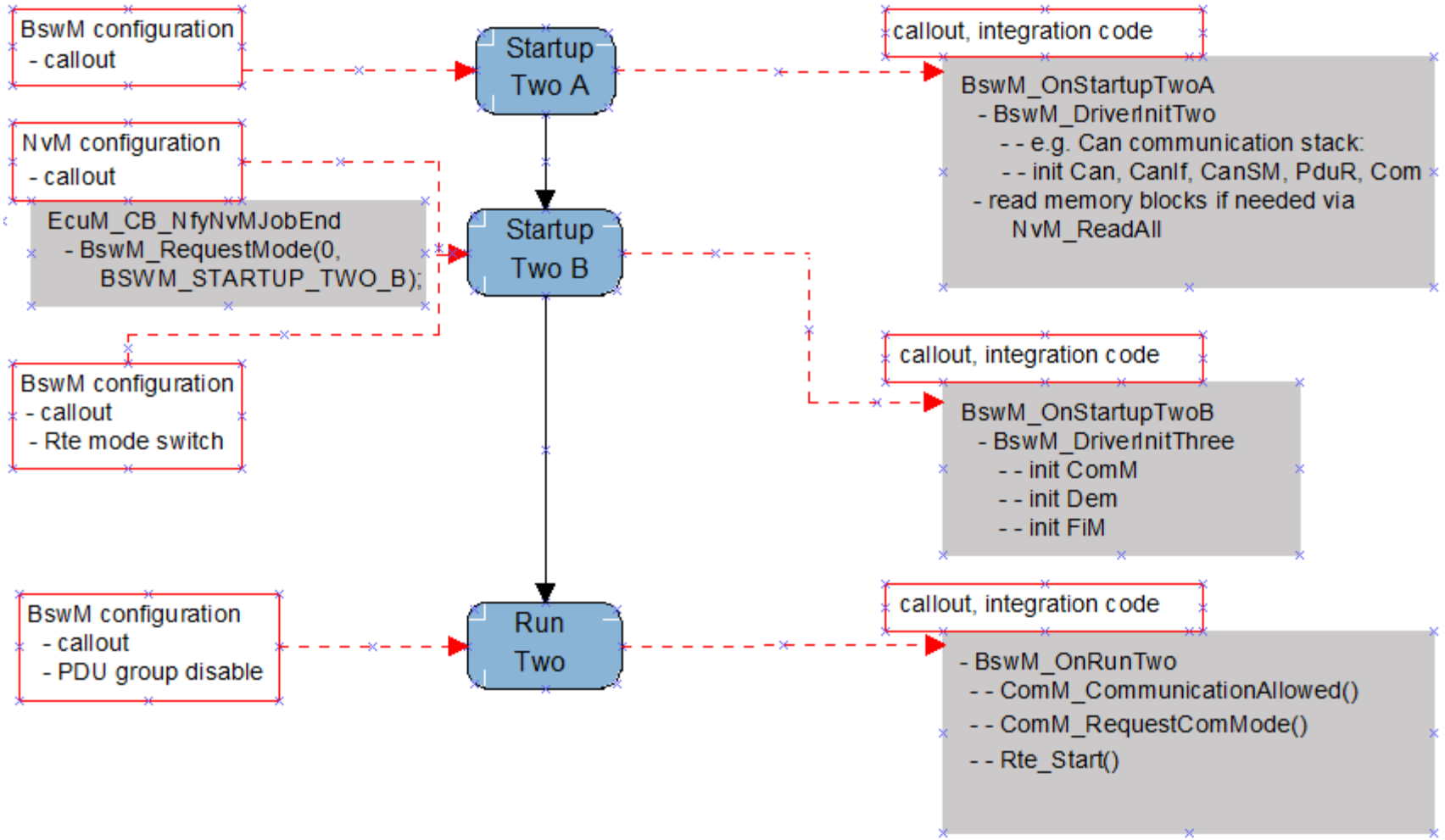
Overview of all states



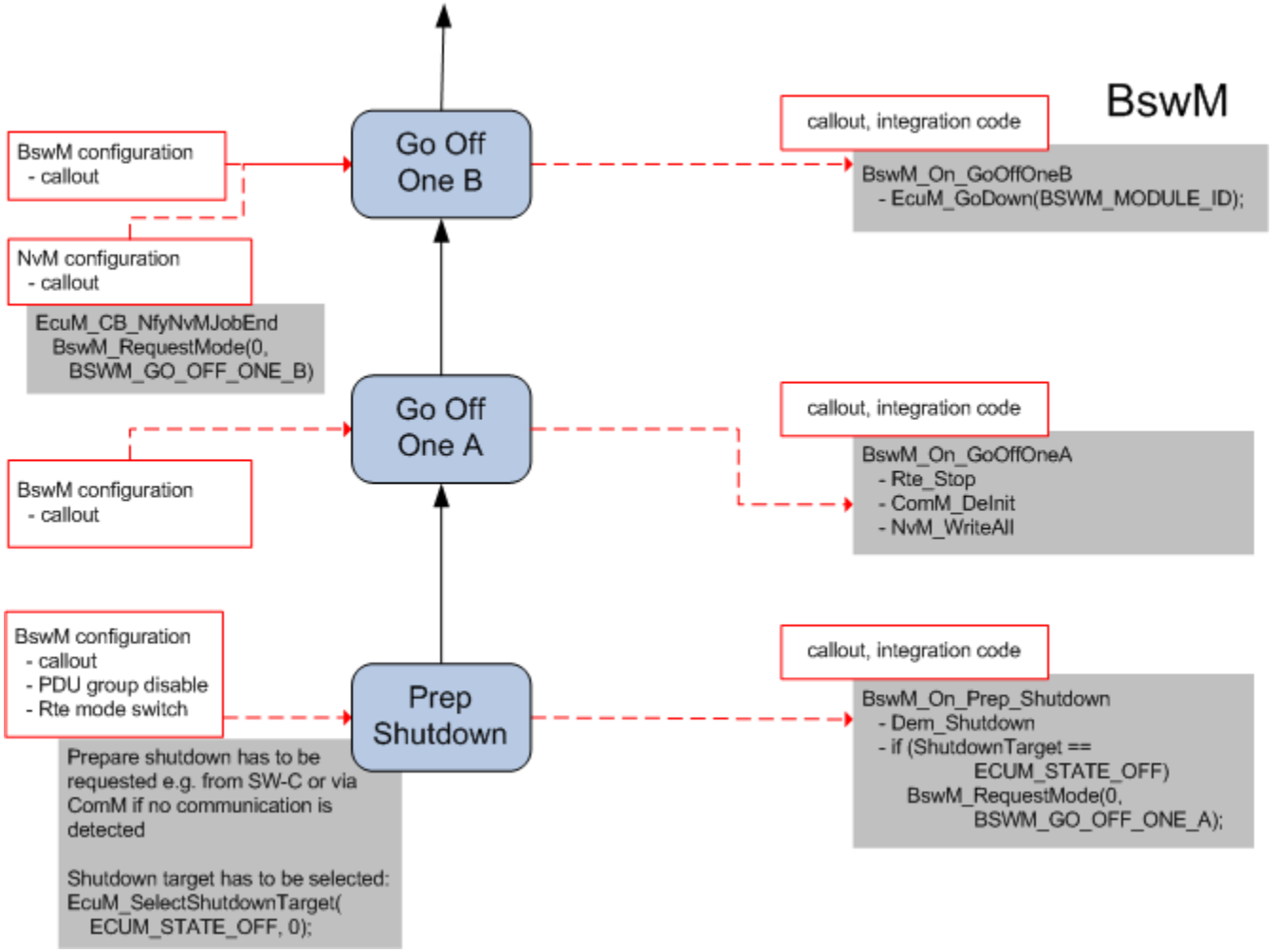
STARTUP - EcuM



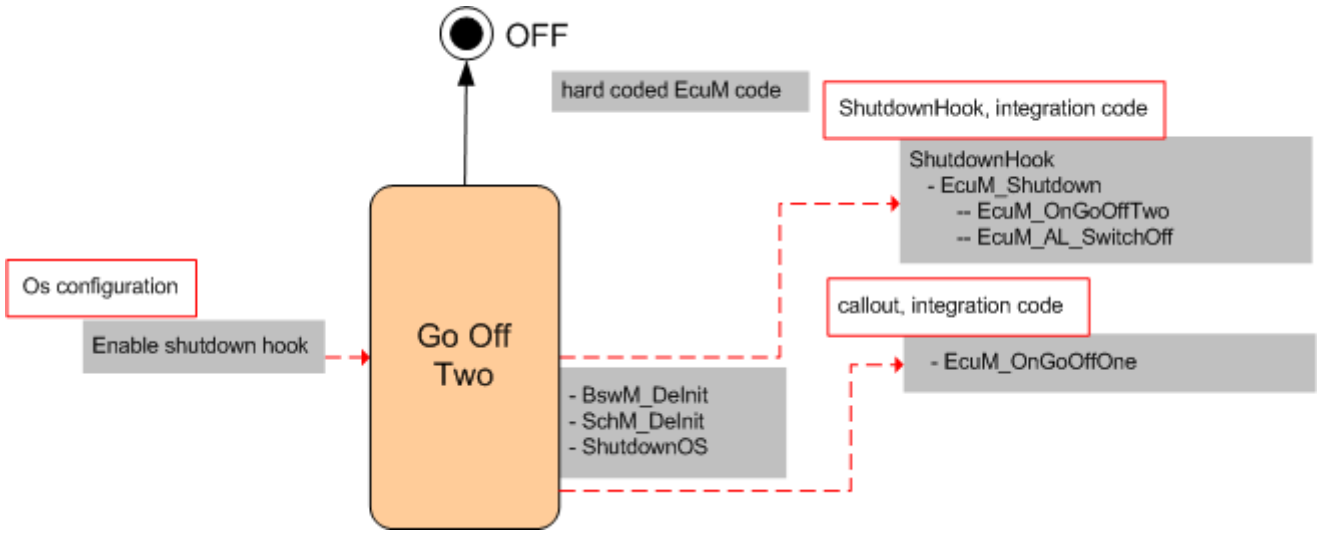
STARTUP - BswM



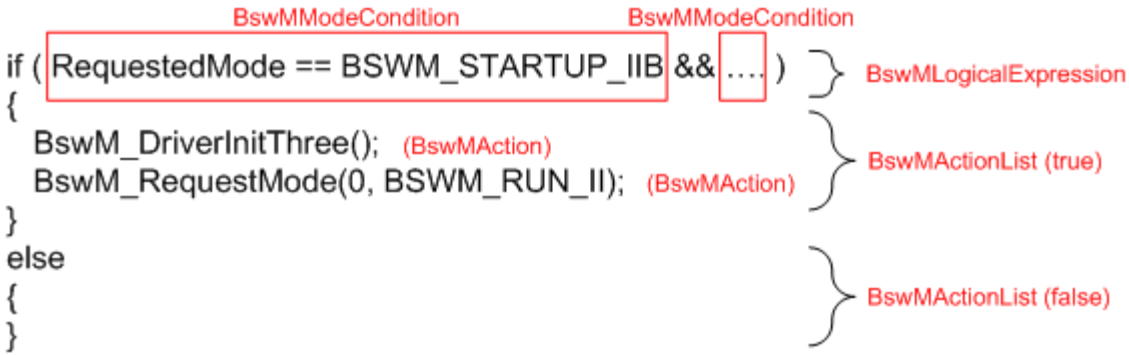
SHUTDOWN - BswM



SHUTDOWN - EcuM



BswM - Rule

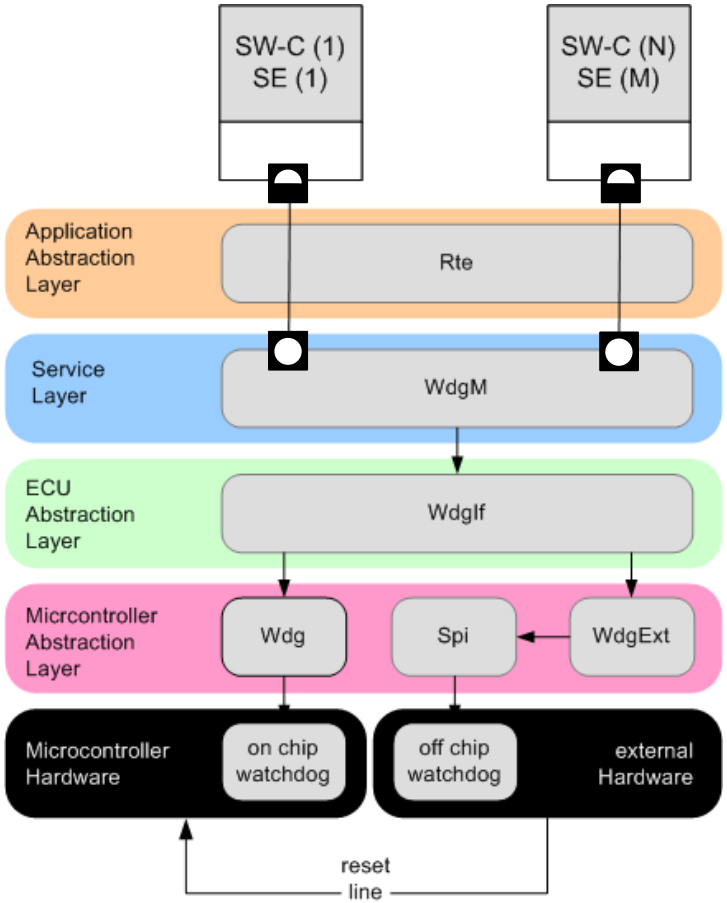


Watchdog stack overview

Trigger Watchdog Driver(s)

Alive Supervision

- Supervision of SWC
- Determination of Global supervision status



ComM overview

- Handles communication modes for each channel
- The COM Manager collects and coordinates the bus communication access requests from communication requestors
- Offer an API to disable sending of signals to prevent the ECU from
- (actively) waking up the communication bus
- Bus error management

Tasks of the Network Management

- Detecting bus activity
  - are other nodes active?
  - Allows vendor-specific extension to identify active nodes
- Synchronizing bus sleep
  - coordination algorithm to ensure that all nodes go to sleep in the same moment*)
- Preventing bus sleep
  - keep the bus (and other nodes) active, while needed
- Coordination of busses
  - synchronize AUTOSAR / OSEK-NM busses
- Least important: Sending (arbitrary) „User Data“

EB tresos classic AUTOSAR training - Memory stack

Elektrobit

Chapter overview

- NVRAM Manager (NvM)
- MemIf
- FLASH EEPROM emulation stack
- EEPROM stack

NVRAM Manager overview

- NVRAM Manager manages all data requests to the EEPROM / Flash used as non-volatile memory

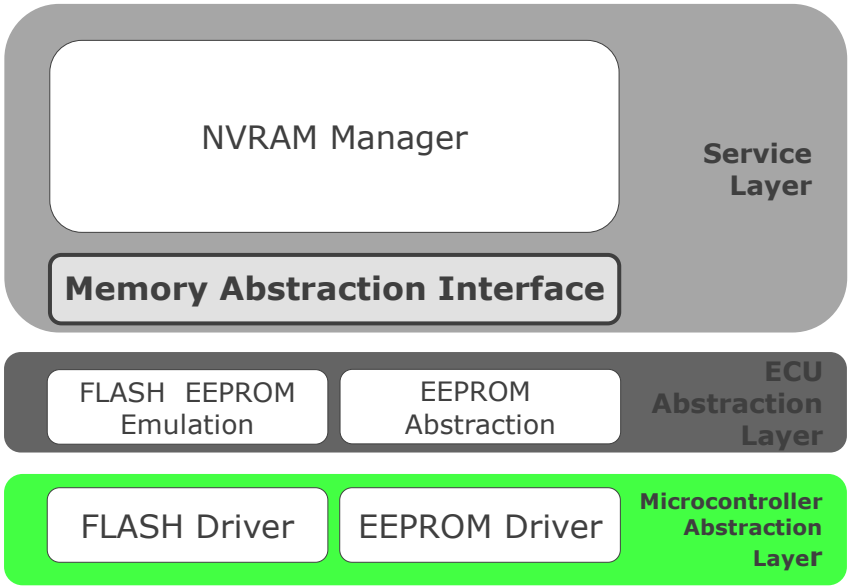
- NVRAM Manager handles async./sync. memory requests like:
  - Read
  - Write
  - Erase

  } asynchronous

  - Validation
  - Status reports
  - Data management settings

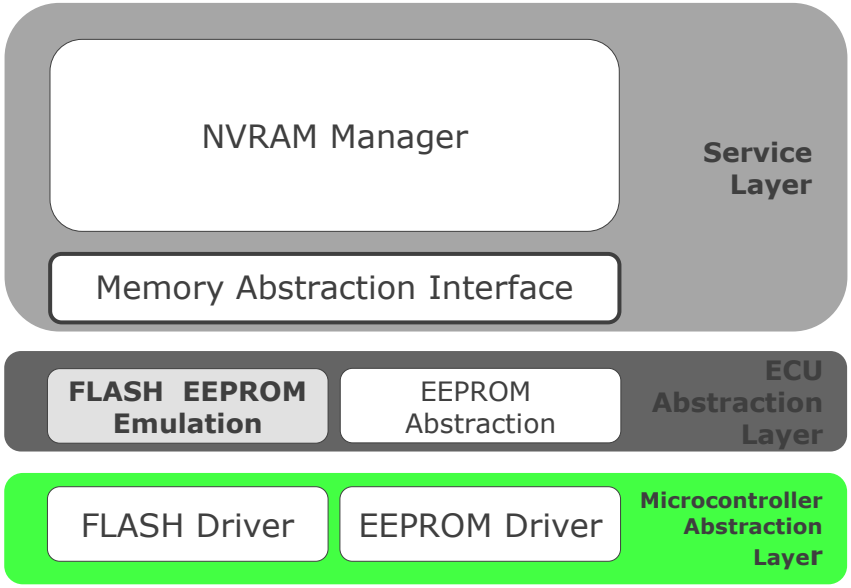
Memory Abstraction Interface - MemIf

- MemIf abstracts the number of underlying memory abstraction modules (Ea, Fee)
- MemIf doesn‘t require any initialization
- MemIf doesn‘t support run-time configuration
- Abstracts function calls to Ea/Fee



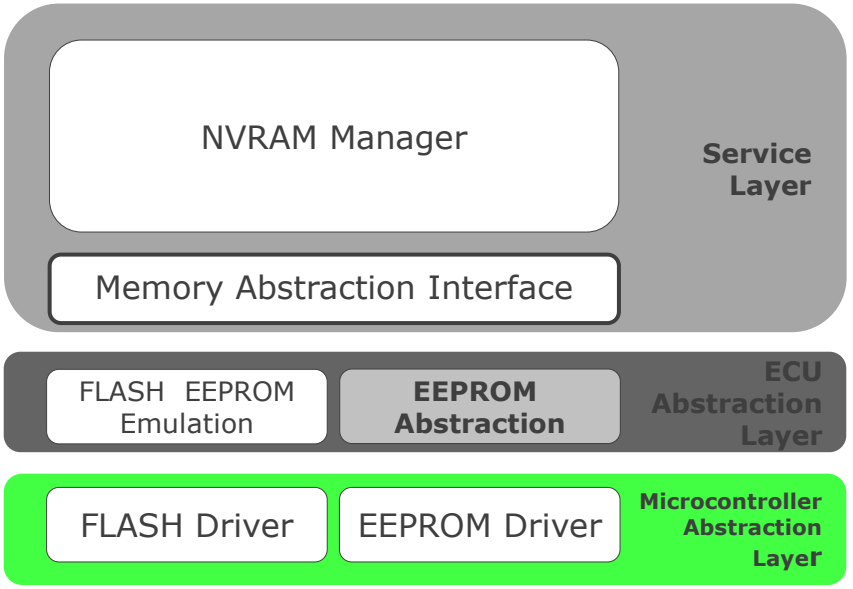
Flash EEPROM emulation (Fee)

- Abstracts device specific addressing scheme and segmentation
- Provides a virtual addressing scheme and segmentation
- Provides a „unlimited“ number of write/erase cycles to the MemIf / NVRAM



EEPROM Abstraction Interface (Ea)

- Manage limitations of erase/write cycles for extended EEPROM write-cycles
- Check for valid data
- Mapping of NVRAM blocks to physical addresses



EB tresos classic AUTOSAR training - Operating system

Main characteristics

- AUTOSAR Os is a static operating system
  - No dynamic handling of system resources
  - AUTOSAR Os is configurable
  - AUTOSAR Os has to be individually configured and generated for each application
- Advantage: AUTOSAR Os can be optimized
- Configuration: EB tresos Studio

Elektrobit

Os Objects related to Scheduling

Tasks

- Tasks are C functions, that can be scheduled by the Os.
- Typically, most parts of the application are executed within a task.

Interrupt Service Routines (ISRs)

- ISRs are handlers for hardware events (interrupts).
- ISRs can interrupt Tasks and possibly even other ISRs.

Events

- Events are an Os mechanism to notify Tasks.
- Tasks can wait for Events.

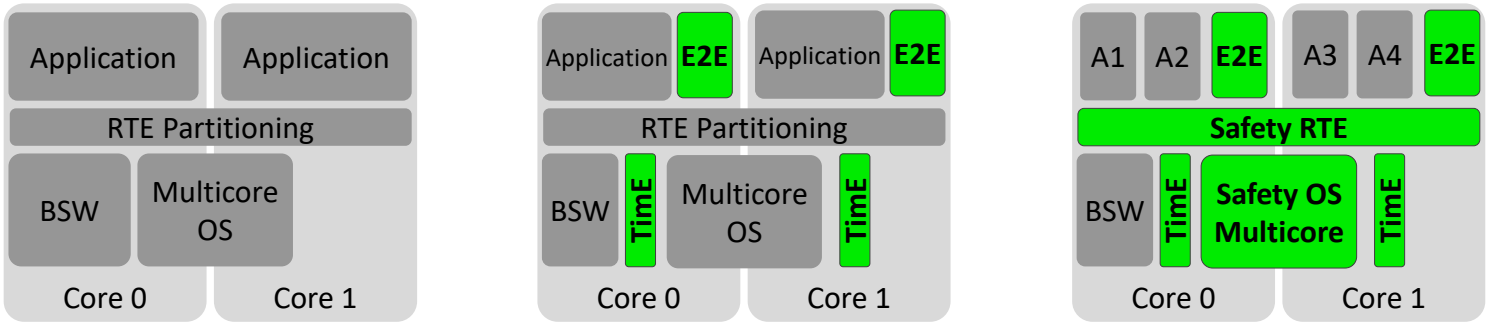
Summary

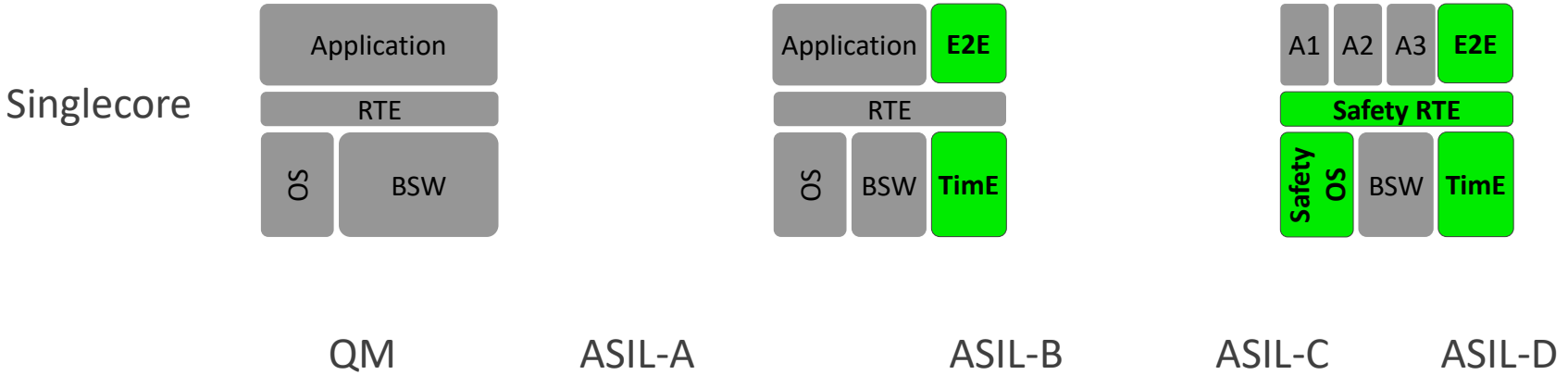
- Tasks are C functions, that can be scheduled concurrently by the AUTOSAR Os.
- There are basic Tasks and extended Tasks. Extended Tasks support waiting for Events.
- Tasks can be preemptive or non-preemptive.
- Task scheduling is based on the priority. For Tasks with the same priority, the Task activation time decides.
- ISRs are handlers for hardware events (interrupts).
- ISRs interrupt the execution of Tasks (unless interrupts are explicitly disabled within the Task).

Usage of
EB tresos Safety Products

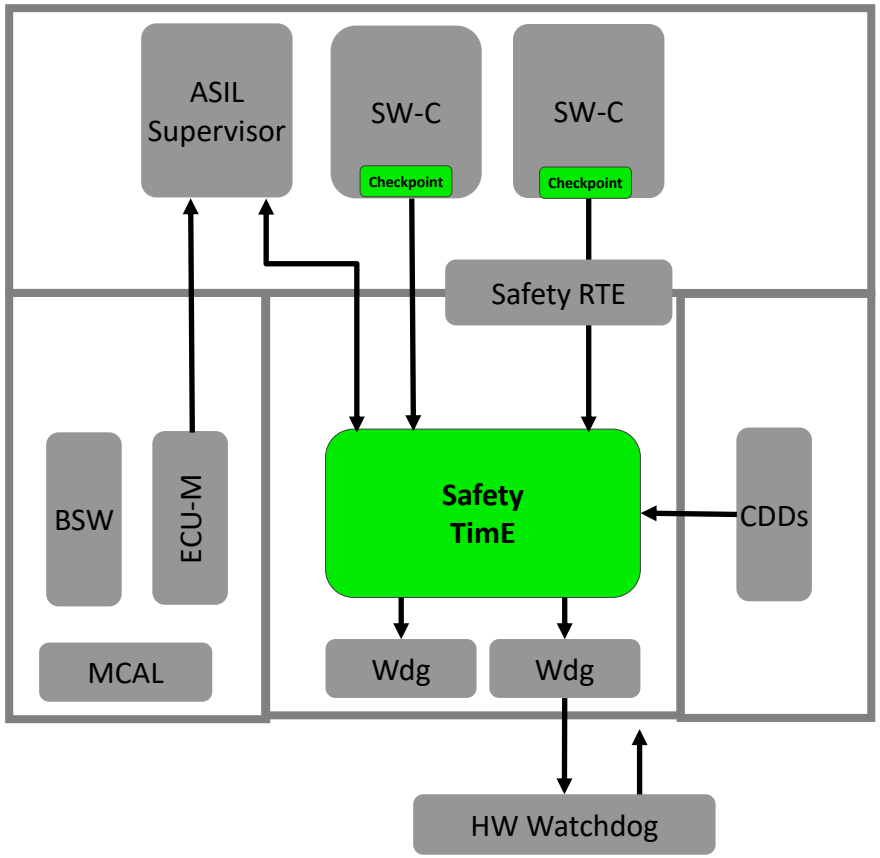
Elektrobit

Scalable Safety Solution



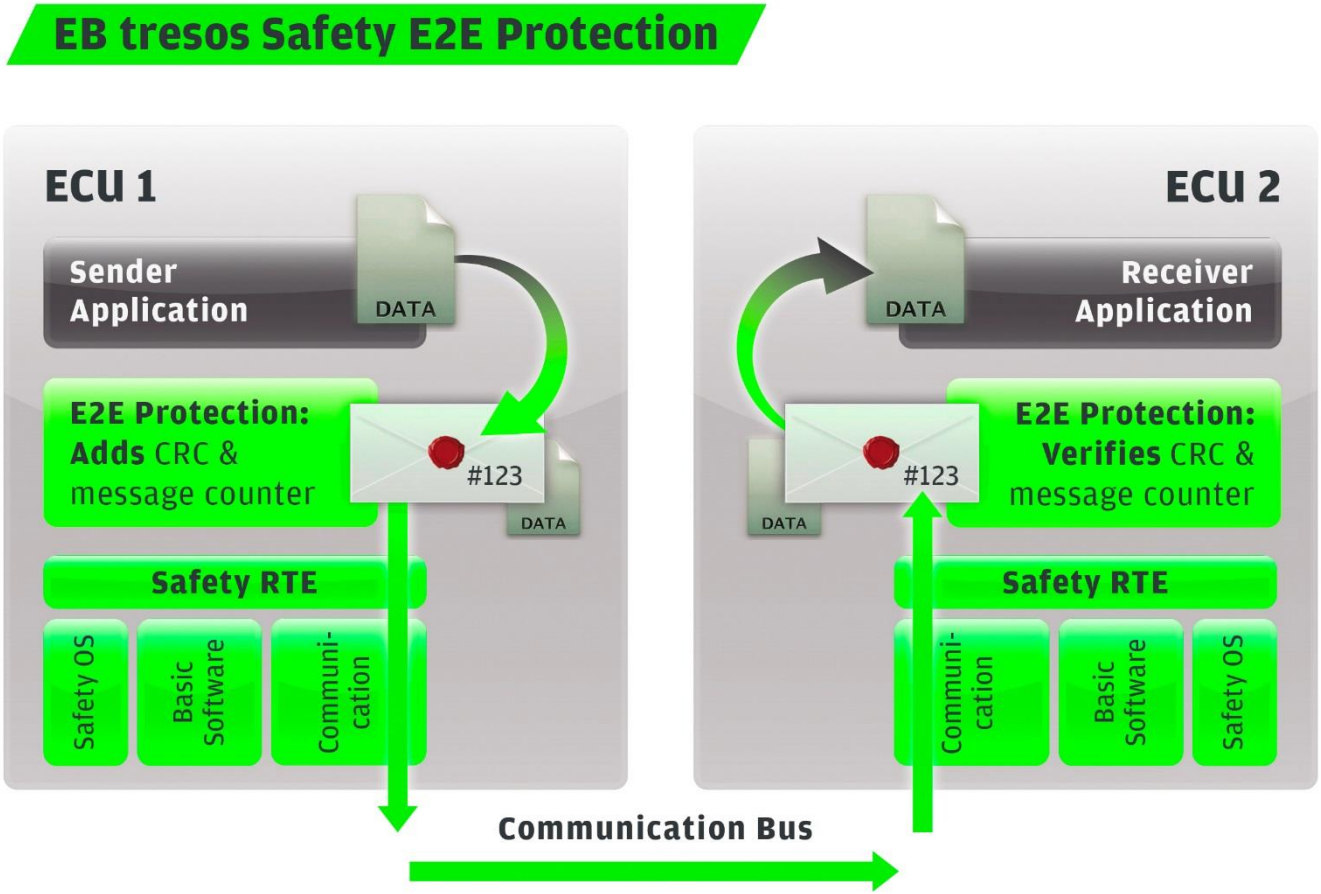


EB tresos Safety TimE Protection



Partitions

EB tresos Safety E2E Protection

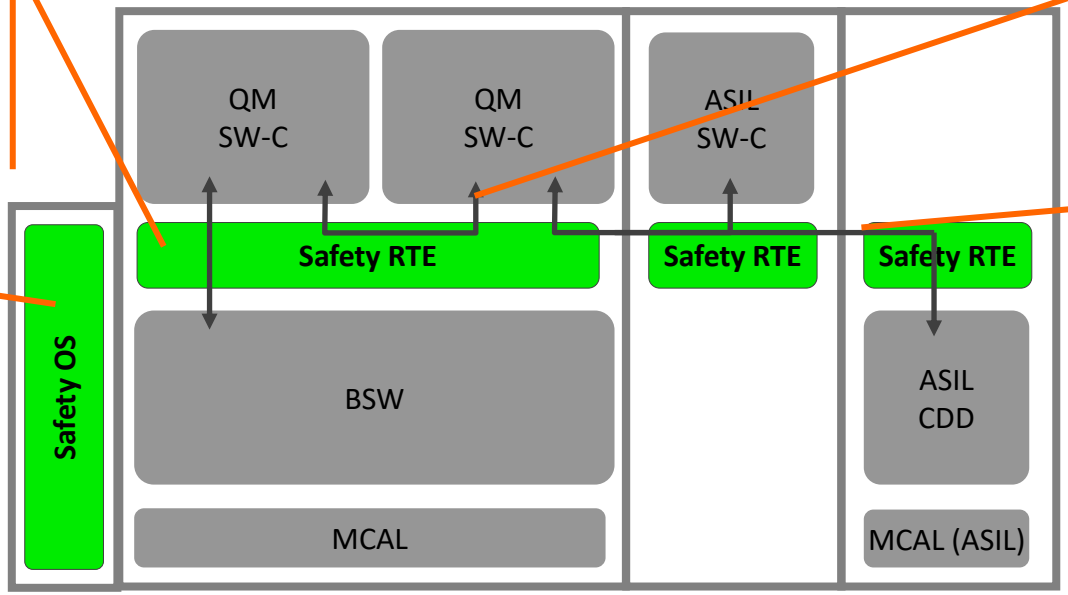


EB tresos Safety OS and Safety RTE

Safety RTE for protected communication between Memory Partitions

Code can be used in ASIL SW-C

Safety OS:
- Data Protection
- Stack Protection
- Context Protection
- OS Protection
- Hardware Error management



Intra Partition communication

Inter Partition communication protected by EB tresos Safety OS

Partitions

Contact us!

Elektrobit

automotive.elektrobit.com