VECTOR

MICROSAR Classic

Product Catalogue

vector.com

Contents

V5.0 [2026-07-20]

1. Overview

The Complete MICROSAR Classic Basic Software Stack

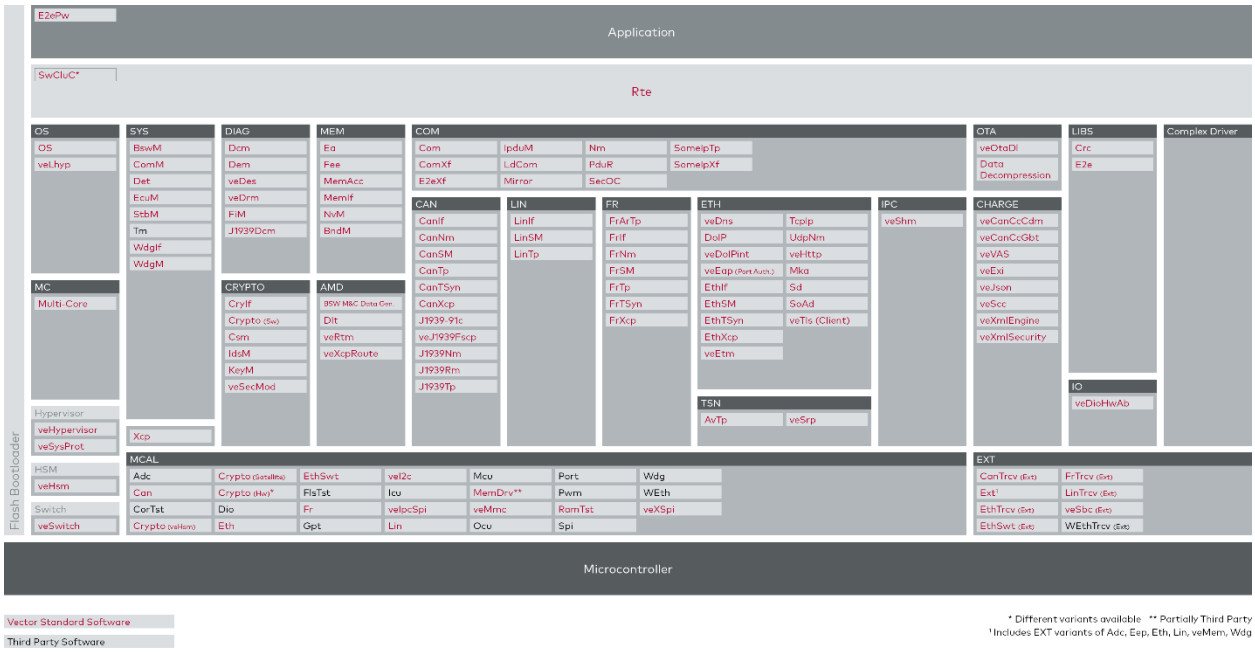

FIGURE 1: MICROSAR Classic BSW stack

MICROSAR Classic by Vector ensures the basic functionality of functional software in ECUs for both single-core and multi-core processors, offering countless functions and many customer-specific modules. All modules can be configured, generated, and integrated according to the project requirements to represent a complete set of ECU software.

> Sophisticated and reliable solution used by leading OEM and TIER1s
> Delivered as turnkey software and preconfigured for specific requirements, optionally delivered as source code
> Available for many hardware platforms and compilers for OS and MCAL modules
> Designed for Automotive, Charging ECUs (GPT, Chademo, Smart Charging), Trucks (OBD2, J1929 Support), Marine (e.g. NMEA2000), medical and special equipment
> Supports many different data formats such as DBC, LDF, FIBEX, CDD, ODX and AUTOSAR-XML
> ASR 4 and ASR 3 compliant from a long-term AUTOSAR Premium member
> Detailed documentation included

For more information about MICROSAR Classic visit vector.com/microsar.

2. OS

Real-Time Operating System for Microcontrollers

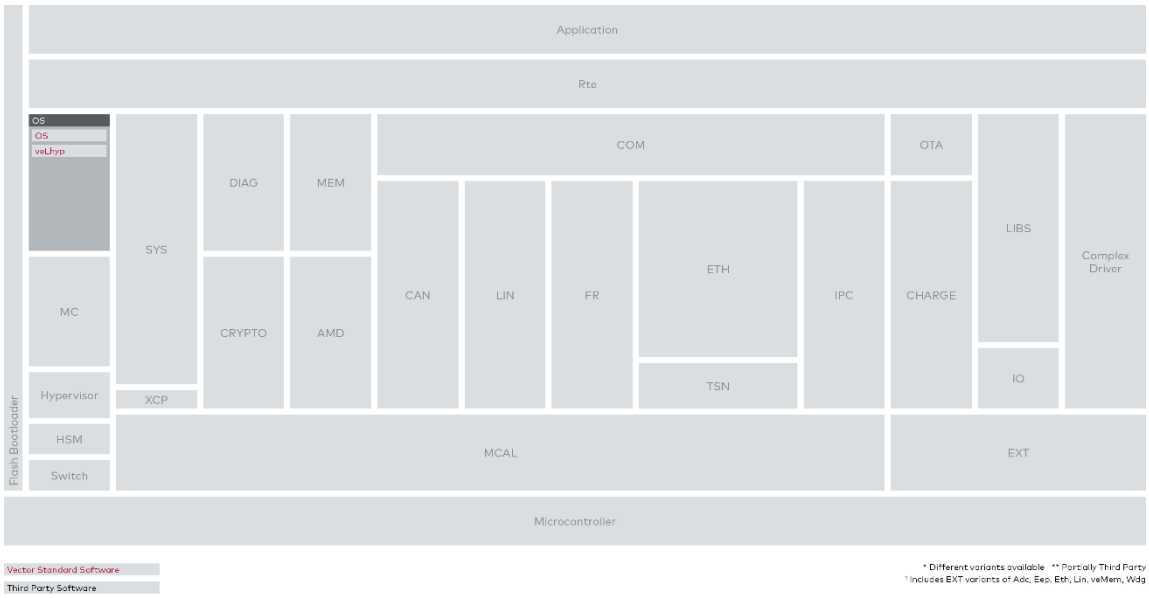

FIGURE 2: OS cluster

OS is a preemptive real-time multitasking operating system, fully compliant with OSEK/VDX-OS and with optimized functions for use on microcontrollers such as time monitoring and memory protection. Protected by the System Memory Protection Unit, different partitions can be operated free of memory access interference, allowing partitions with different ASIL to run in parallel. The Vector Lean Hypervisor (veLhyp) ensures safe startup of multiple operating system partitions in a multi-core processor or SoC and initializes the system MPU.

> Small, robust, quick, and resource-economizing operating system and with short boot times
> Graphic configuration tool for easy configuration
> Timing resolutions below 1ms without increasing interrupt load (High-Resolution Timer)
> Possible resolutions in microsecond range (hardware dependent)
> Available for many 16-, 32- and 64-bit microcontrollers and as multi-core operating system
> Safe start-up of multiple operating system partitions
> Implementation according to ISO 26262 ASIL D
> Freely selectable master core

2.1 Operating System (Os)

The OS module is an extended OSEK operating system for ECUs (also available as GuestOS Linux and QNX) which supports multi-core microcontrollers and can be run in safety-related environments. The variations are divided into Scalability Classes (SC), offering Schedule Tables, Timing Protection (time synchronization to measure and monitor tasks and interrupt service routines) and Memory Protection (mechanisms for easier and more reliable integration of applications).

> SC1: Schedule Tables
> SC2: Schedule Tables and Timing Protection
> SC3: Schedule Tables and Memory Protection
> SC4: Schedule Tables, Timing Protection and Memory Protection

Add-Ons

> Multi-Core (symmetric): For multi-core systems with a same set of commands
> Multi-Core (asymmetric): For multi-core system with a different set of commands

2.2 Vector Lean Hypervisor (veLhyp)

The Lean Hypervisor module is developed according to ISO 26262 ASIL D as Safety Element out of Context (SEooC). It programs the system MPU at system startup, starts the operating system partitions and subsequently no longer requires a CPU load. In a multicore application, veLhyp initializes the System Memory Protection Unit (MPU) at startup and manages the startup of the cores. Each core can have its own operating system image, where any combination of POSIX, classical or adaptive AUTOSAR operating systems is conceivable. The master core is freely selectable. If the hardware does not boot an ASIL compliant kernel after a reset, protection initialization can be assigned to another ASIL compliant kernel.

veLhyp makes partitions independent. Each partition can focus on its local memory protection needs. In addition, partitions do not need to implement a synchronized wait state for basic initialization. This reduction of external dependencies facilitates the development of the individual partition.

3. MC

Multi-Core for Basic Software and RTE

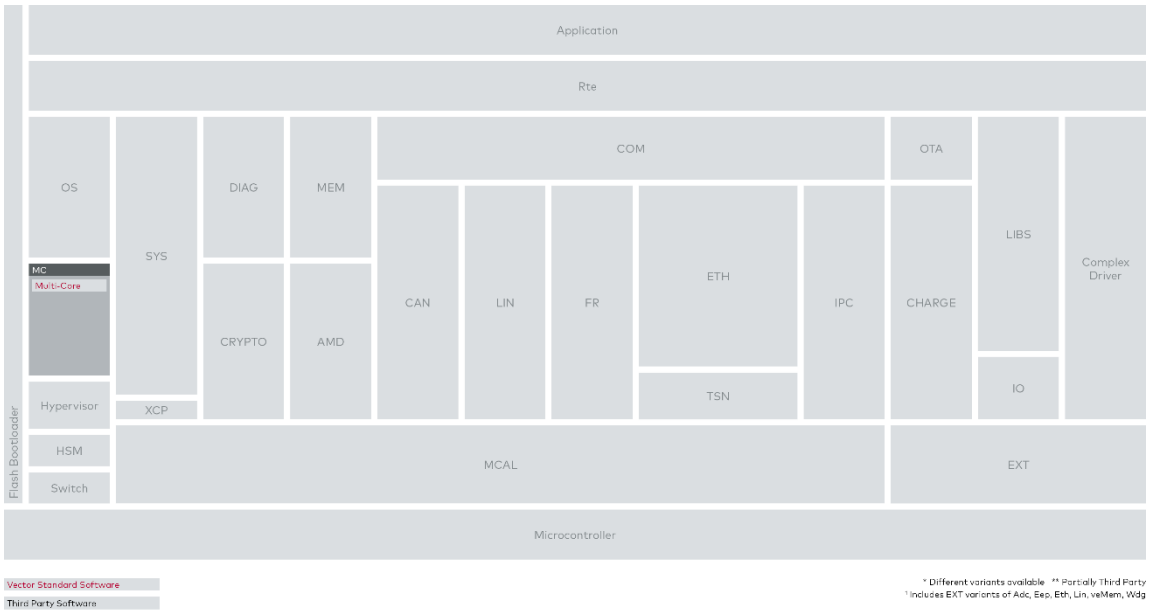

FIGURE 3: MC cluster

3.1 Multi-Core

Multi-Core allows to distribute the application specific workload to different cores. and Application Software Components to different physical cores or OS partitions of the same OS instance – achieved by RTE cross-core communication and dedicated Basic Software satellites that provide a high-performance solution for interfacing Basic Software features from different cores.

Add-Ons

> MixedASIL & BSW-Split: This add-on allows to distribute the processing of complete communication stacks (e.g. CAN or ETH) to different cores with the goal to offload stack specific tasks to different cores. Using integrated cross core communication, it is possible to access signals and PDUs also from applications being executed on other cores. Additionally, this concept allows mapping of communications stacks to different OS partitions with varying safety level, e.g. to allow coexistence of ASIL B ETH stack in an overall ASIL D setup.

4. COM

Network Independent Communication and Gateways

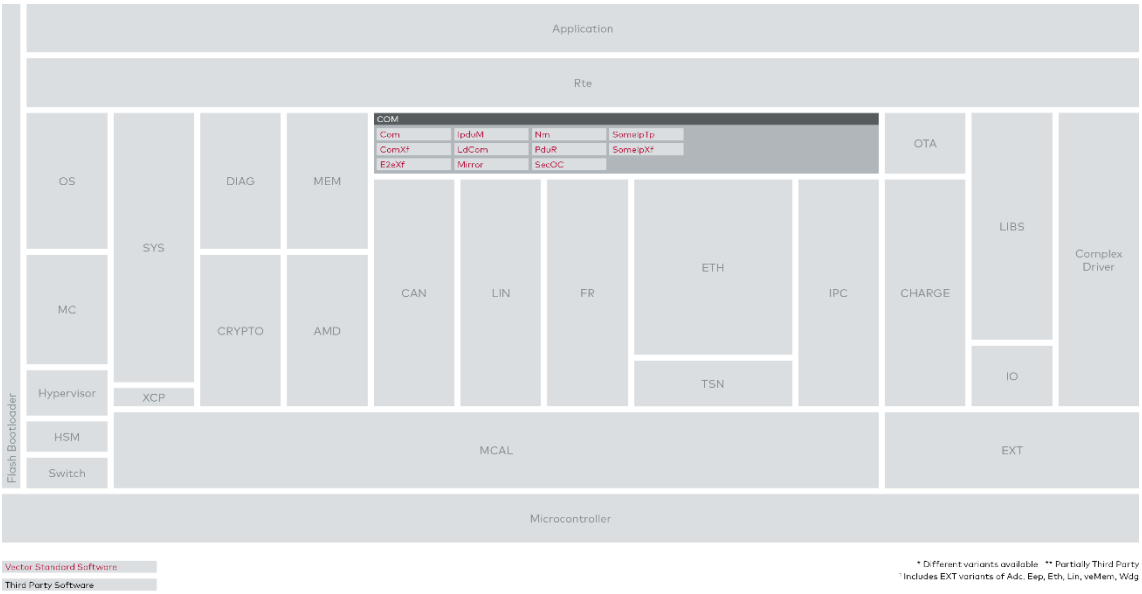

FIGURE 4: COM cluster

COM contains services for ECU communication that support any number of communication channels and allows completely bus-independent software development required in any communication stack, whether for CAN, CAN-FD, J1939, FR, LIN or ETH. All tasks for transmitting messages and for cross-bus network management activities are handled by the configurable Basic Software modules. No additional software is required for a gateway ECU. The modules Com and PduR enable routing of signals and TP or application messages.

> Code and execution time optimized by application-specific configuration
> Extended support for Nm coordinators
> Network management module: OSEK Nm compatibility is configurable
> Simultaneous operation of different Network Management Interfaces (AUTOSAR/OSEK Nm) in migration projects

4.1 Communication (Com)

The Com module provides a signal-based data interface for the RTE and organizes the transmitting of messages according to their transmission type (e.g. cyclic or event triggered). The module contains various message mechanisms for receiving signals and manages initial values, update bits and timeouts on the signal level. In multi-channel ECUs, the integrated signal gateway offers the possibility to route signals between the communication buses. Furthermore, it enables bus-independent signals of the functional software to be implemented in PDUs.

> Invalidity declaration of Tx signals in case of Rx signal timeout
> Optimizations of runtime reduction of main functions (Rx-sided: caching of received events, Tx-sided: configuration of multiple time domains)
> Deferred Event Caching of Rx IPDUs: This is an optimization of the Rx main function run-time via an event triggered processing of Rx PDUs. The optimization allows to waive cyclic polling of all PDUs in the main function Rx.

Add-Ons

> Gateway: The Com module can be supplied with gateway functionality. The Routing is possible for signals, signal groups or via a configuration description without existence of a real signal or signal group.

> HighEndFeatures: A powerful alternative to signal-based routing and cycle-delay PDU routing is the description-based routing, which enables the routing of PDU segments (defined by start bit and length) including the emission class (periodic, event-driven, on changes). This function is available for all COM modules and requires the gateway option GW for Com.

4.2 veGw

The Vector Gateway offers a high-performance routing solution running on dedicated gateway cores and uses a strict polling-based apporach (includes vGwM and veGw-Task).

4.3 COM Transformer (ComXf)

The ComXf module allows efficient transmission of complex data structures and big PDUs over the network with many group signals. The placement is derived from the system extract.

4.4 E2E Transformer (E2eXf)

The E2eXf module enables end-to-end protection for network communication (e.g. serialization through ComXf or SomeIpXf).

4.5 I-PDU Multiplexer (IpduM)

The optional I-PDU Multiplexer (IpduM) module supports multiple usage of frames with different data contents, via a static configuration for the classic bus systems or alternatively via dynamic data content mapping for CAN FD.

4.6 Large Data Communication (LdCom)

The LdCom module optimizes the routing of large signals by avoiding unnecessary copying of data.

4.7 Gateway Mirroring (Mirror)

The Mirror module allows mirroring internal busses to the diagnostic access which enables to read normally inaccessible messages on the bus and to identify problems. This functionality allows to

> mirror one internal CAN or LIN channel to diagnostic CAN
> mirror multiple CAN, LIN, FR and ETH channels to diagnostic ETH
> mirror FR channels to diagnostic ETH

4.8 Network Management (Nm)

The Network Management Interface (Nm) bundles inter-bus network management activities of all the ECU's communication channels. As network management coordinator, it synchronizes wake-up and sleep of the communication channels.

> Synchronous sleep and wake-up of multiple networks via different Nm coordinators
> Backup coordinator
> Support of OSEK Nm (configurable)
> Mixed operation of OSEK and AUTOSAR Nm on one channel

4.9 PDU Router (PduR)

The PduR module provides an interface to the communication modules (interface, transport protocol and network manage-ment) of the different bus systems for the modules Com, Dcm and the complex drivers. This interface serves to transmit and receive data via PDUs. The PduR also implements a gateway between the communication modules of the various bus systems. The module Cdd allows TP- and IF-PDUs to be integrated into the COM stack (above/below the PDU router or above the communication interfaces).

Add-Ons

> Gateway: Allows TP and message routing. The routing is possible via metadata in case of range routing, of variable addresses (dynamic gateway) or of dynamic PDU lengths.

4.10 Secured Onboard Communication (SecOC)

The SecOC module is used to send or receive authenticated messages between two ECUs. Unauthorized, repeated or manipulated messages are detected. SecOC interacts with the PDU router, can be controlled by the application and is part of the MICROSAR security solution. A Freshness Value counter is used to prevent replay attacks. It is generated by the independent component Freshness Value Manager (FVM). Different approaches are supported for generating the Freshness Values.

> Transmission of authenticated and integrity-protected I-PDU's
> Authentication with a Message Authentication Code (MAC). The actual generation and validation of the Message Authentication Code is performed by the CSM.

4.11 SOME/IP Transport Protocol (SomeIpTp)

The SomeIpTp module offers a transport protocol for the Scalable Service-Oriented Middleware over IP (SOME/IP) which is typically used for a segmentation of large messages.

4.12 SOME/IP Transformer (SomeIpXf)

The SomeIpXf module provides a serialization strategy for various data types. LdCom can be used here for a highly efficient transmission.

4.13 veGw

Vector Gateway. High-performance routing solution running on dedicated gw cores and uses a strict polling-based apporach (includes vGwM and veGw-Task).

4.14 veGwHwRM

Vector Gateway Hardware Routing Manager. Allows to extract possible routings and setup of a routing table. Enables configuration of hardware-specific routing modules like drivers or CDDs (routing engines).

5. CAN

CAN Communication

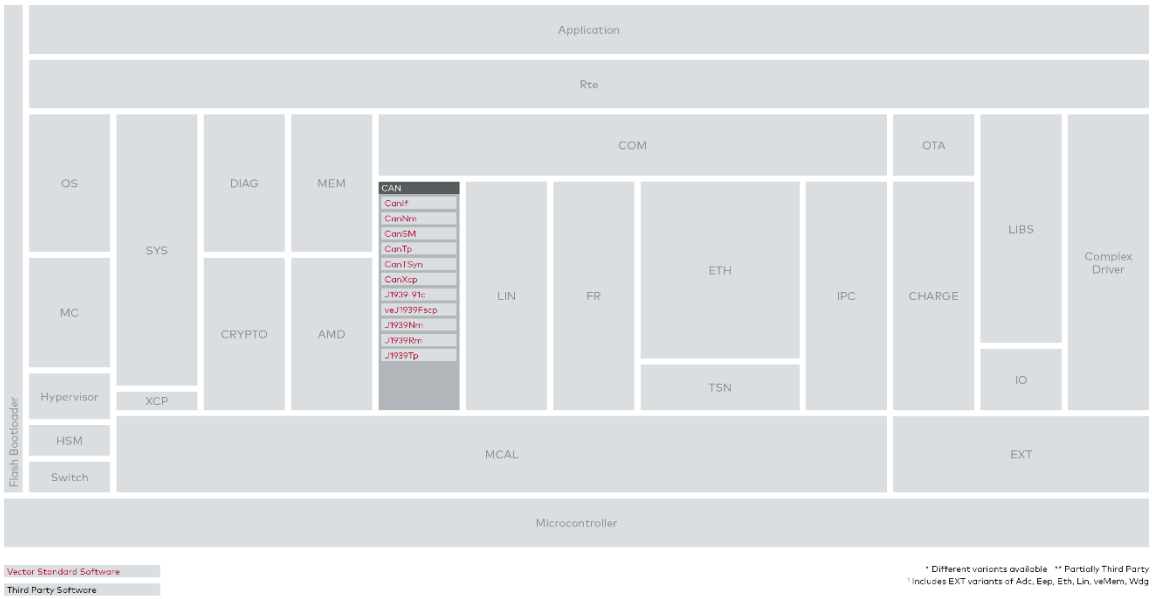

FIGURE 5: CAN Cluster

CAN contains the Basic Software modules for CAN communication and optional modules for J1939 (designed for heavy-duty vehicles with transport protocols BAM and CMDT) – suitable as a basis for calibration with XCP, gateways or reprogramming.

> Code and execution time are optimized based on need-specific configuration
> Inter-module configuration of all communication-specific software modules
> Fast wake-up handling at ECU start-up
> ISO 15765-2 compatibility is configurable (CanTp)
> OSEK Nm is available as a configurable module, compliant with CanNm
> On-off control of the communication stack depending on the partial network state (CanNm, CanSm)
> Support of up to 64 Byte data with enhanced bandwidth, available for many CAN FD controllers

5.1 CAN Interface (CanIf)

The module CanIf offers abstracted (PDU-based) access to the CAN Driver. It controls the CAN Driver (Can) as well as the transceiver driver (CanTrcv).

> Double hash search algorithm for efficient filtering the Rx messages

Add-Ons

> CAN-XL is a continued development of CAN FD and provides vehicle-wide communication with same semantic regardless of physical connection (CAN XL / Ethernet) or communication paradigm (Signal-and Service-based communication) offering the following main features:
  > Payload size up to 2048 Byte
  > Net bitrate up to 20 Mbit/s
  > Support of complex network topologies
> CAN-XL Ethernet tunneling: This feature provides support for tunneling of IEEE 802.3 frames.

5.2 CAN Network Management (CanNm)

Within a CAN network, the CanNm module is responsible for the coordinated transitions between the wake-up and sleep states.

> Precompile optimizations, e.g., for single channel systems

5.3 CAN Station Manager (CanSM)

The module CanSM is responsible for the bus-specific error handling.
> Support of ECU Passive Mode

5.4 CAN Transport Protocol (CanTp)

The CanTp module conforms to ISO standard 15765-2. As a transport protocol for CAN, it is responsible for segmenting the data in the Tx direction, collecting the data in the Rx direction, and monitoring the data stream.
> Precompile optimizations, e.g., for single channel systems
> Supports mixed addressing (11-bit CAN ID), typically for CAN/LIN gateway applications
> Optimized routing (e.g., with Burst Transmission) together with the PduR from COM
> ISO 15765-2 compatibility is configurable

5.5 CAN Time Synchronization (CanTSyn)

The CanTSyn module implements the CAN-specific time synchronization protocol. Access to the synchronized time base by the SWCs requires the Synchronized Time-Base Manager (StbM).
> Time Synchronization over CAN (CanTSyn) implements the time synchronization protocol for CAN. This enables clock synchronization between CAN ECUs as part of the AUTOSAR Global Time Concept. The Synchronized Time Base Manager module (StbM) of SYS is available as higher-level time coordinator.

5.6 CAN XCP Module (CanXcp)

Xcp is a protocol for communication between a master (PC tool) and a slave (ECU). It is standardized by ASAM and is mainly used for measuring, calibrating, flashing, and testing ECUs. XCP supports the bus systems CAN (CanXcp), FlexRay (FrXcp) and Ethernet (EthXcp).

5.7 CAN-FD Network Security for J1939 (J1939-91C)

J1939-91C enables secure on-board communications for J1939 on CAN-FD (J1939-22) with support of full J1939-91C compliant behavior of the security tag (32-bit FV, 1 E-bit, 31-bit CMAC), including handling of multiple FvM instances and full Rekeying support.

5.8 Functional Safety Communication Protocol for J1939 (veJ1939Fscp)

The SAE J1939-76 functional safety communication protocol is a part of the J1939 standard which supports safety-relevant projects by a safety header message including sequence counter and CRC to validate the content of the subsequent J1939 parameter group.

5.9 Network Management for J1939 (J1939Nm)

The J1939Nm module handles the Address Claiming and NAME management on J1939 networks as specified in SAE J1939-81, which ensures unique source addresses and node identification on a J1939 bus. J1939Nm does not support sleep and wake-up handling, CanNm can be used for that purpose.
> Fully dynamic addressing: automatic change of own address (in case of conflict) and automatic adaptation of all used addresses to the current changes

Add-Ons
> Dynamic Nm: J1939Nm uses fully dynamic addressing on defined buses. It can change the ECUs own addresses (in case of conflict) and handle address changes from other ECUs accordingly. This supports adding of ECUs to networks at runtime, and is required for ISO 11783 (ISOBUS).

5.10 Request Manager for J1939 (J1939Rm)

The module J1939Rm implements requesting of data via Request Handling that is defined in the SAE J1939 protocol.

5.11 Transport Layer for J1939 (J1939Tp)

The J1939Tp module handles the transport protocols BAM (Broadcast Announce Message) and CMDT (Connection Mode Data Transfer) as specified in SAE J1939-21 and their CAN FD variants of SAE J1939-22. CMDT is also known as RTS/CTS.

> Adds transport protocols ETP (ISO11783-3, ISO11783-7) and FastPacket (NMEA200)

Add-Ons

> ISOBUS: Adds the Extended TP (ETP) of ISO11783-3 and ISO11783-7 and the Fast Packet Protocol (FPP) of NMEA2000.

6. FR

FlexRay Communication

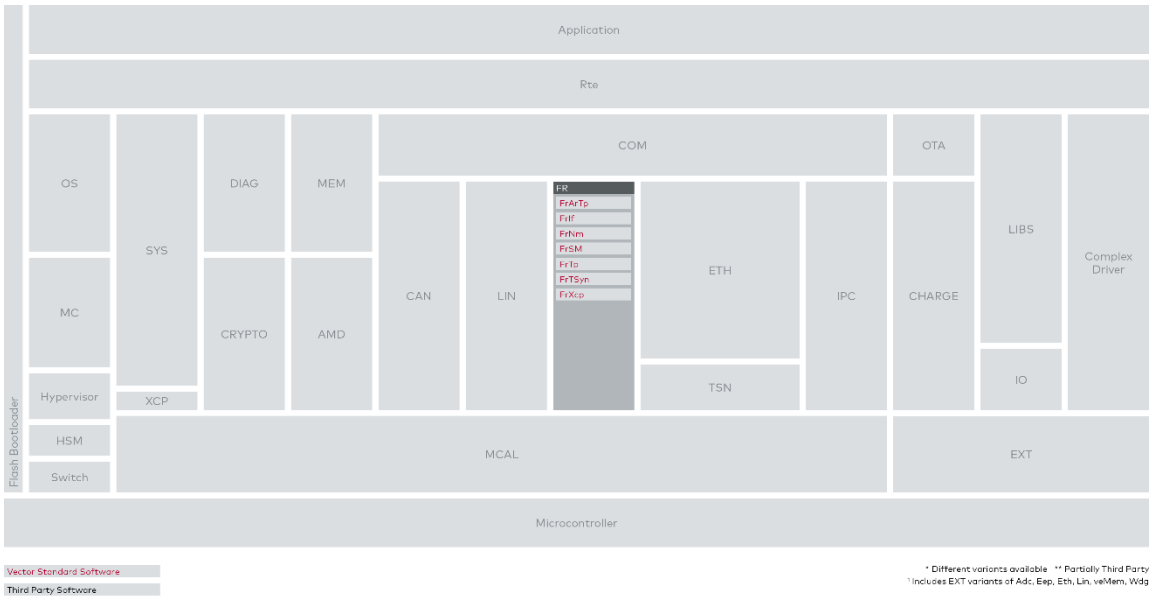

FIGURE 6: FR cluster

FR is used in FlexRay networks or sub-networks and is ideal as a basis for calibration with XCP, gateways or flashing. FlexRay Basic Software modules can be used without an operating system. However, it is recommended to use MICROSAR Classic OS (ideal for FlexRay applications), a conventional OSEK OS (e.g. Vector osCAN) or an AUTOSAR OS.

> Activates and deactivates partial networks and provides data, depending on the partial network state
> Small code size and short execution times due to optimized administration of job lists of the FlexRay interfaces
> Usage of either FrTp (AUTOSAR) or FrIsoTp (ISO 10681) transport protocol
> Support of ECU passive mode in the FlexRay State Manager
> Early detection of synchronization losses

6.1 FlexRay AUTOSAR Tranport Protocol (FrArTp)

FlexRay transport protocol based on ISO 15765-2 (CanTp). It contains a frame compatibility with the CAN bus.

6.2 FlexRay Transport Protocol (FrTp)

FrTp is a FlexRay transport protocol and is based on the ISO 10681-2 standard.

> Precompile optimizations, e.g., for single channel systems

6.3 FlexRay Interface (FrIf)

The module FrIf offers abstracted (PDU-based) access to the FlexRay hardware and the support for synchronization with the global FlexRay time.

> Support of the APIs: CancelTransmit and L-PDU reconfiguration
> Dual channel redundancy for redundant transmission of frames and PDU-specific voting function for the SWCs
> Precompile optimizations, e.g., for single channel systems

6.4 FlexRay Network Management (FrNm)

The module FrNm is responsible for FlexRay network management. It synchronizes the transition to the bus sleep state.

> Precompile optimizations, e.g., for single channel systems

6.5 FlexRay State Manager (FrSM)

The module FrSM controls and monitors the wake-up and start-up of nodes in the FlexRay cluster.

> Support of ECU Passive Mode, immediate start-up after passive wake-up, extended error handling by State Change Notification, configurable time delay for FlexRay start-up at passive wake-up as well as configurable number of wake-up patterns.

6.6 Time Sync Over FlexRay (FrTSyn)

The module FrTSyn realizes the FlexRay specific time synchronization protocol. An access to the synchronized time base by the SWCs requires the Synchronized Time-Base Manager (StbM).

> Time Synchronization over FlexRay (FRTSYN) implements the time synchronization protocol for FlexRay. This permits clock synchronization between FlexRay ECUs as a part of the AUTOSAR Global Time Concept. The Synchronized Time Base Manager module (StbM) from SYS is available as a higher-level time coordinator.

6.7 FlexRay XCP Module (FrXcp)

The FrXcp module contains FlexRay-specific contents of the XCP module (Xcp).

7. LIN

LIN Communication

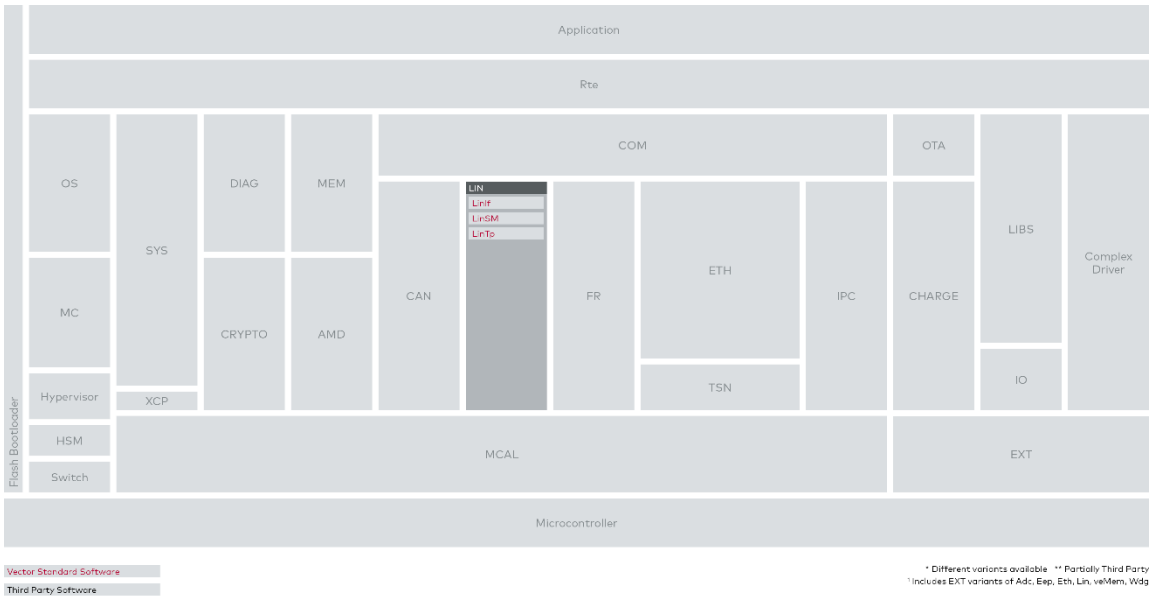

FIGURE 7: LIN cluster

LIN serves communication tasks for a Commander or Responder in a LIN network. In addition, it can be used as a basis for XCP, gateways or reprogramming. XCP is also available as an ASAM extension for the LIN Commander.

> Minimized scheduling jitter for multi-channel master
> Optimized routing of diagnostic requests to LIN Slaves
> Quick start of the LIN channel
> Reliable switching of schedule tables possible
> Based on Vector's many years of experience with production software for LIN

7.1 LIN Interface (LinIf)

The module LinIf offers abstracted (PDU-based) access to the LIN hardware. It also handles schedule table processing. It is available as master or slave.
Configurable wake-up delay
Separately configurable memory mapping of configuration data for LinIf and LinTp. This is especially appealing for controllers with segmented memory.

> Notification of Schedule Table End
> Configurable schedule tables to reduce maximum task runtime in multi-channel systems
> Wakeup through LIN transceiver. After an external wake-up, this function allows to omit a second (unwanted) wake-up pulse by the master.

7.2 LIN State Manager (LinSM)

The module LinSM switches between schedule tables and PDU groups in the Com module and services the LIN interface regarding sleep and wake-up. This module is available as master and slave.

> Extended polling of LinSM sub-modes for controlled switchover of LIN Schedule Tables
> Optimized start-up behavior by automatic selection of a schedule table (configurable)

7.3 LIN Transport Protocol (LinTp)

The LinTp module, available as commander and responder, is responsible for segmenting the data in the Tx direction, collecting the data in the Rx direction and monitoring the data stream. LinTp is a component of LinIf and is offered as an option, as a transport protocol is not usually mandatory in the LIN communication stack.

Add-Ons

> AutoAddressPidAssignment: LIN Slave Node Position Detection method according to the LIN Bus Shunt Specification is supported with optional PID assignment.

8. ETH

Ethernet-Based Communication

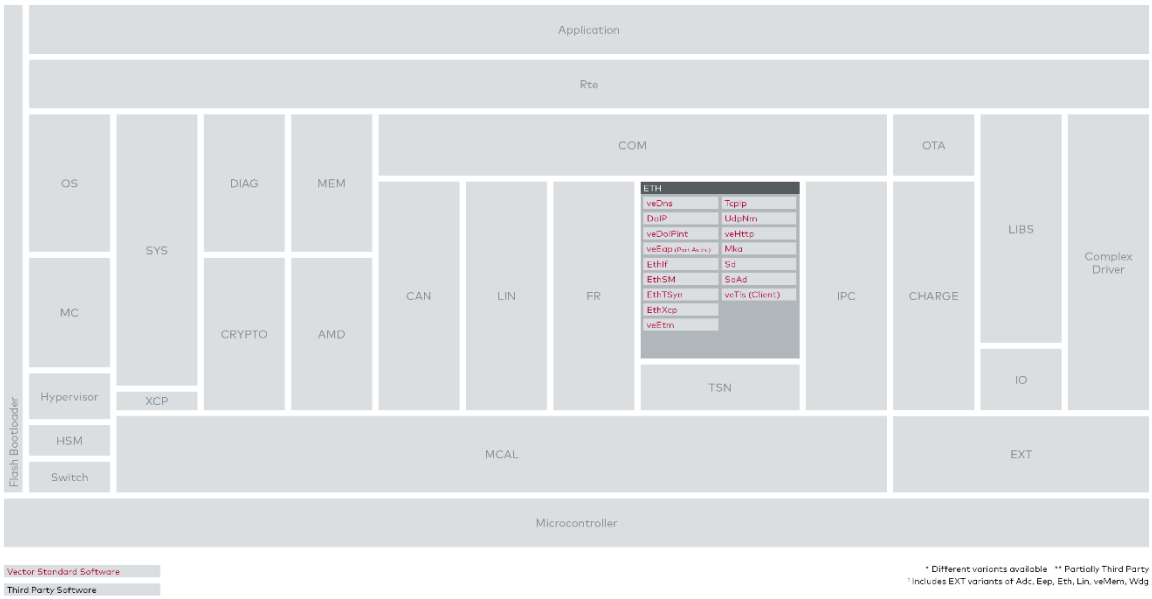

FIGURE 8: ETH cluster

ETH provides diagnostics, measurement, and calibration with Ethernet-based communication within a vehicle or between external infrastructure and the vehicle – signal and PDU-based, via UDP and TCP. Ethernet significantly accelerates software download and diagnostics of Ethernet ECUs. Gateways can be implemented to route diagnostic requests to in-vehicle networks and reprogram multiple CAN ECUs in parallel via DoIP (with Flash Bootloader directly via DoIP). XCP is available for measurement and calibration with the advantage of a larger bandwidth and expands the gateway possibility to calibrate CAN and FlexRay ECUs via the Ethernet port. With SOME/IP and Service Discovery (SD), data can be transmitted in a service-oriented and service-managed manner. In addition, SOME/IP provides dynamic serialization of data.

> TCP/IP Stack developed according to an automotive standard
> IETF conformance regularly proven via OPEN ALLIANCE TC8
> No open-source software
> Seamless integration of vehicle-to-grid communication and A/V bridging into Ethernet and TCP/IP Stack
> Ethernet switch configuration and time synchronization between ECUs
> Seamless integration of vehicle-to-grid communication and audio/video bridging into the stack configuration and time synchronization between ECUs
> Easy integration of custom functions/modules at all levels
> Diagnosis of the vehicle according to ISO 13400-2 (DoIP)
> Fast and parallel reprogramming of ECUs with a conventional PC or diagnostic tester

8.1 Vector Domain Name System Resolver (veDns)

The veDns module contains a DNS Resolver. It is responsible for resolving a domain, e.g., vector.com, into a valid IP address.

8.2 Diagnostic over IP (DoIP)

The DoIP module provides diagnostic functionality after ISO 13400-2. Requires SoAd.

8.3 Vector Diagnostic over IP for vehicle internal communication (veDoIPint)

veDoIPint implements a transport protocol for diagnostic data via IP. It is derived from the DoIP module according to ISO 13400 and adapted for in-vehicle communication. Unlike the DoIP, where communication is always initiated by the tester and the ECU waits for the incoming data, the veDoIPint module can take both roles. As a result, it can be used in any node of a vehicle. However, the use of veDoIPint requires DoIP and SoAd.

8.4 Ethernet Interface (EthIf)

The Ethernet interface provides hardware-independent access to the Ethernet driver (EthDrv), Ethernet transceiver driver (EthTrcv) and Ethernet switch drivers (EthSwtDrv and EthSwtDrv EXT). It is also responsible for VLAN handling.

8.5 Ethernet State Manager (EthSM)

To start up or shut down communication in Ethernet clusters, the Ethernet State Manager (EthSs) provides an abstract interface to the Communication Manager (ComM). The EthSm accesses the Ethernet hardware over the EthIf.

8.6 Time Sync Over Ethernet (EthTSyn)

Time Synchronization over Ethernet (EthTSyn) implements the time synchronization protocol for Ethernet and permits clock synchronization between Ethernet ECUs. The Synchronized Time Base Manager (StbM) module from SYS is available as a higher-level time coordinator.

> Initialization of all necessary hardware interfaces
> Ethernet frames with Ethertype 0x88F7 are routed from the ETHIF to the ETHTSYN
> Support of gPTP (Generalized Precision Time Protocol) plus enhancements
> Support of General and Event messages
> Support of calculation of propagation time delays (Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up)
> Support of transport of the synchronous time stamp (Sync, Follow_Up)
> Based on AUTOSAR/IEEE 802.1AS with additional Automotive relevant features
> Part of AUTOSAR Global Time Concept

8.7 XCP over Ethernet (EthXcp)

The EthXcp module contains Ethernet-specific contents of the Xcp module.

8.8 Vector Ethernet Testability Module (veEtm)

veEtm represents an upper tester and implements the AUTOSAR testability protocol which is needed to perform protocol conformance tests. The module enables an externally connected test environment to trigger defined actions, e.g., sending UDP packages or creating a TCP connection.

8.9 Vector Hypertext Transfer Protocol (veHttp)

One application of the Hypertext Transfer Protocol is to transmit browser requests to a server. The module contains an HTTP Client.

8.10 Vector MACsec Key Agreement (veMka)

The veMka module contains the MACsec key agreement function (supporting EthIf, Eap and Eap-Tls).

8.11 Vector MACsec over Software veMACsec (SW)

This module allows to realize MACsec in software.

8.12 Service Discovery (Sd)

Service Discovery (Sd) is a module through which an ECU informs its communication partners which services are offered. ECUs can also register to receive automatic notifications, for example when a signal is updated.

8.13 Socket Adaptor (SoAd)

The SoAd module converts communication via PDUs into socket-oriented communication and implements XCP routing (ISO 13400-2 diagnostic functionality is in the DoIP module).

Add-On

> BSD: Enables the SoAd and the overlying modules to be used in a non-AUTOSAR environment like LINUX

8.14 Transmission Control Protocol/Internet Protocol (TcpIp)

The TcpIp module contains all protocols for UDP/TCP based communication and supports IPv4 and IPv6 and parallel operation of IPv4 and IPv6 in one ECU (for a working TcpIp at least one add-ons is required).

Add-Ons

> TcpIp v4: IPv4, ICMPv4, ARP and DHCPv4 (Client)
> TcpIp v6: IPv6, ICMPv6, NDP and DHCPv6 (Client)
> DHCP v4 Server: Assigns IP addresses based on the switch port
> IPsec (Internet Protocol Security): Allows to establish an IPsec communication according to IETF RFC 4301. The functionality is restricted to transport mode and the usage of Authentication Header only according to RFC 4302. The Authentication Header adds data integrity and data authentication to the payload but does not allow confidentially.
> TIs

8.15 UDP Network Management (UdpNm)

The UdpNm module controls the synchronous sleep of Ethernet ECUs by network management via UDP.

8.16 Vector Transport Layer Security (veTIs - Client)

The veTIs module contains a Transport Layer Security client and encrypts the TCP-based communication (encryption algorithm selectable). The use of veTIs (client) is limited to smart charging.

8.17Vector Switch Host (veSwitchHost)

This module allows the MICROSAR BSW to exchange information with MICROSAR Switch.

8.18 Udp-over-Ip v4 (TcpIpRel)

Reduced UDP subset to enable use of IPv4 in fail-operational systems. This item is available only in a High Availability-Use Case.

9. TSN

Time Sensitive Networking

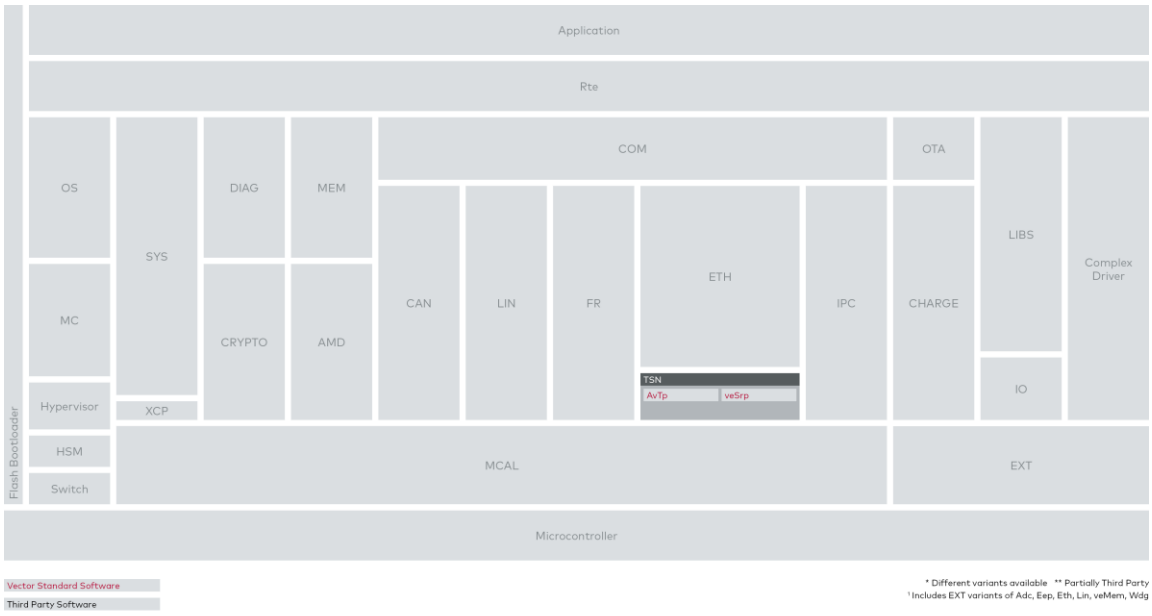

FIGURE 9: TSN cluster

9.1 Audio/Video Transport Protocol (AvTp)

The module AvTp is specified in IEEE 1722. In AVB networks it is responsible for the transport of audio/video data, including the Presentation Time.

> Interface to the EthIf module for receiving and sending AVTP frames
> Distinction made between Stream and Control channels
> Display and validation of the time stamp
> Detection of the transported data stream

9.2 Stream Registration Protocol (veSrp Bridge)

The veSrp Bridge module (stream registration with admission control for multiple streams from different end stations) ensures that the total reserved bandwidth for an Ethernet AVB network does not exceed a predefined limit.

10. IPC

Inter-Processor-Communication

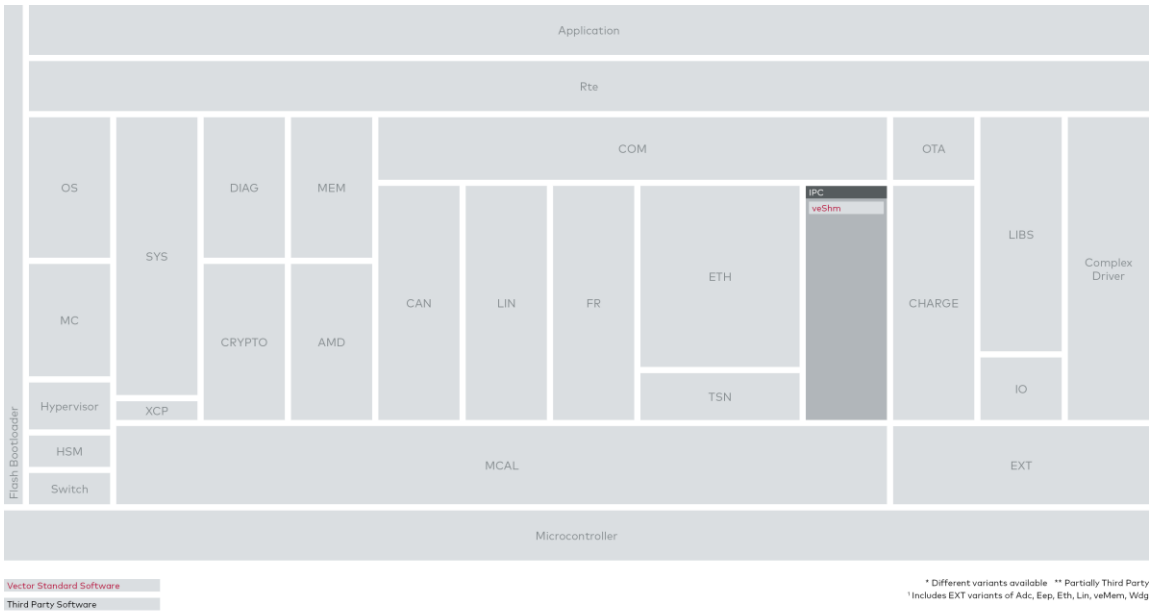

FIGURE 10: IPC cluster

Vector's IPC solution provides message-based communication between different embedded software stacks (operating systems) on the same ECU. The provided mechanism can be used to communicate across microcontrollers, processors or cores on a shared hardware platform.

> Shared memory or SPI are used for exchanging PDUs
> Supports communication between different embedded software stacks such as MICROSAR Classic and MICROSAR Adaptive
> Supports forwarding of bus traffic (e.g. CAN) from one operating system to a different operating system without bus access
> Supports communication between microcontrollers on a multi-microcontroller ECU
> Supports communication between microcontroller and microprocessor on same ECU
> Support of service-oriented and signal-based communication

10.1 IPC via shared Memory (veShm)

The veShm module is used to exchange PDUs via shared memory. Dedicated RAM sections are configured to be abstracted as shared memory channels and are used as ring buffer to exchange PDUs. The component IpduM (I-PDU Multiplexer) is used to multiplex a range of messages to one shared memory channel and can be used to assign SOME/IP headers to PDUs.

11. CHARGE

Communication with External Charging Infrastructure

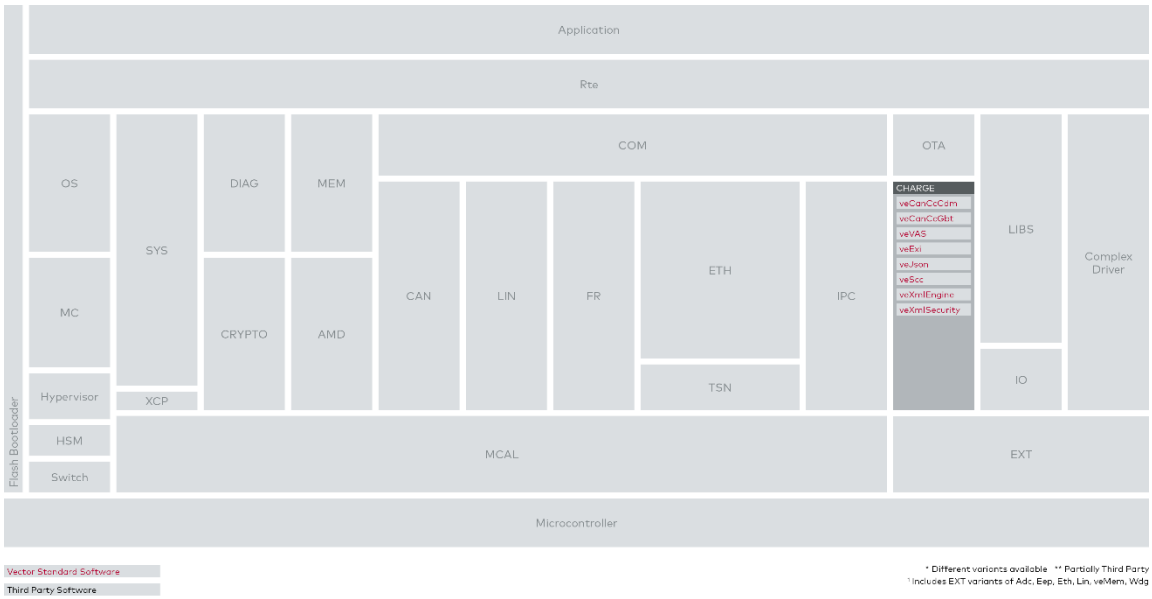

FIGURE 11: CHARGE cluster

CHARGE contains Basic Software modules for intelligent charging of electric and hybrid vehicles and for communication with the infrastructure via Internet technologies such as HTTP. The modules can also be used to connect the ECU to a server via common Internet protocols including TLS. It supports ISO 15118, DIN SPEC 70121, CHAdeMO and GB/T 27930 with options AC/DC, Wireless Power Transfer (WPT), Automatic Charging Device (ACD) and Bidirectional Power Transfer (BPT).

> Implements all protocols needed for Smart Charging Communication (veScc)
> Supports communication via internet mechanisms and protocols
> Easy integration of customer specific functions through generic interfaces

11.1 Vector Efficient XML Interchange (veExi)

The veExi module is used to interpret XML documents and convert them to binary format. This makes processing and transmission of the files more efficient, which economizes communication bandwidth.

11.2 Vector JSON Parser (veJson)

Contains a parser for JSON, which is a JavaScript-based data exchange format and can be used instead of XML.

11.3 Vector Smart Charging Communication (veScc)

This module is responsible for Smart Charging Communication according to ISO 15118 and DIN SPEC 70121. Vector provides the application interface as specified by the customer as a add-on.

Add-Ons

> AC: AC charging
> DC: DC charging
> ACD: Automatic charging device (pantograph charging)
> WPT: Wireless power transfer (inductive charging)
> BPT: Bidirectional power transfer (energy recovery)
> OPP: Opportunity Charging (OppCharge)

> ESDP/ENP: Extensible SECC Discovery Protocol (ESDP) and Event Notification Protocol (ENP)

11.4 Vector XML Engine (veXmlEngine)

The veXmlEngine module contains a parser for processing and a generator for creating valid XML 1.0 documents (used in the CHARGE field).

11.5 Vector XML Security (veXmlSecurity)

The veXmlSecurity module is used to generate and validate XML signatures to EXI-encoded data (based on the W3C XML security standard). It is used in the Vehicle2Grid environment ("Plug and Charge" according to ISO 15118).

11.6 Vector CAN Charging Communication CHAdeMO (veCanCcCdm)

This module contains the standardized charging communication for direct current charging according to the specification CHAdeMO.

11.7 Vector CAN Charging Communication GB/T 27930 (veCanCcGbt)

This module contains the standardized charging communication for direct current charging according to the specification GB/T 27930 for China, available in the variants veCanCcGbt (201x) and veCanCcGbt (202x). Support for diagnostics for this standard can be developed on request.

11.8 Vector Value-added Services (veVAS)

The veVAS module supports the Fire-Preventing Charger Battery Information Exchange Protocol as specified by the Korea Environment Cooperation. It includes an application interface to get the battery data as well as means to control/monitor the TCP/IP connection.

12. MEM

Managing Nonvolatile Memory

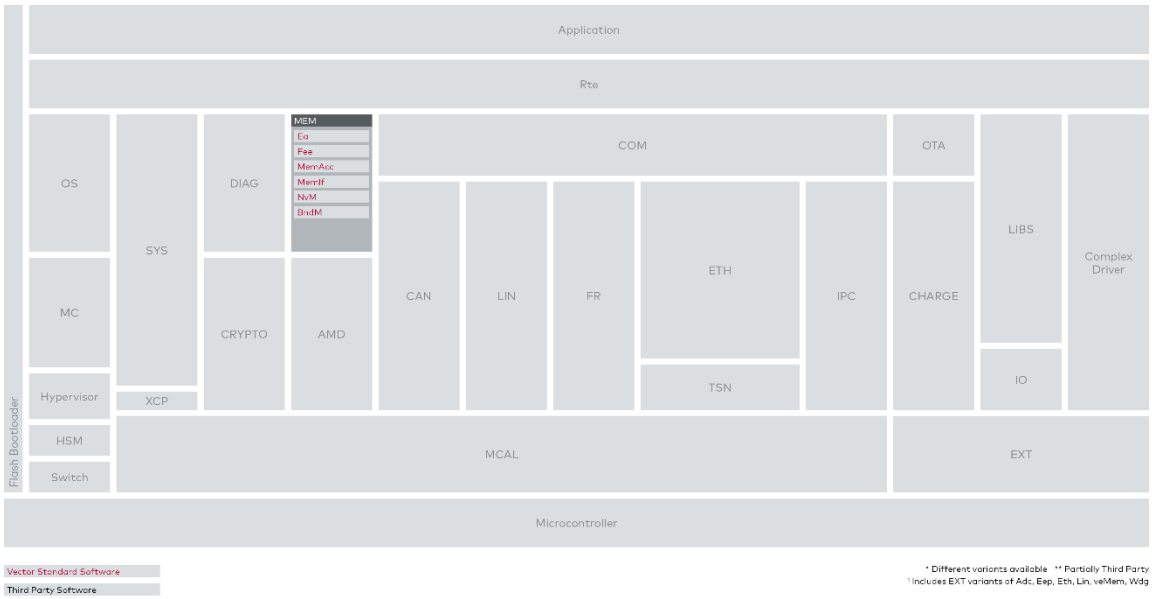

FIGURE 12: MEM cluster

MEM includes services for reading, writing, and deleting persistent application data in nonvolatile memory (flash or EEPROM) – fast, reliable, and robust memory management for managing, checking in and restoring data. This gives functional software access to memory regardless of the specific type of memory present on the platform and whether that memory is internal to the controller or externally connected.

> Particularly secure data transactions and efficient data access
> Efficient and robust management of nonvolatile memory
> Redundant storage of management data increases the reliability of data access
> Cross-module configuration of the entire memory stack
> Platform-optimized memory stack
> Solution from a single source
> Mixing several flashing and EEPROM chips in one ECU is possible
> Depending on the use case, the memory stack requires additional Basic Software modules
> For special requirements, Vector offers platform-optimized solutions
> Platform-optimized memory stack solutions
> For special requirements, Vector offers platform-optimized solutions

12.1 EEPROM Abstraction (Ea)

The Ea module offers a hardware-independent interface for accessing EEPROM data. In addition to reading, writing, and clearing data, the Ea module also distributes write accesses to different areas of the EEPROM, so that all EEPROM cells are uniformly stressed, which increases their lifetime.

> Additional configurable transaction security, which is a standard feature of the Fee module
> Redundant storage of management data for increased reliability of data access

12.2Flash EEPROM Emulation (Fee)

The Fee (FlexNor) module provides a universal interface for managing flash memory. It allows reading, writing, and erasing data, and effectively spreads write operations across different memory regions to enhance the lifespan of flash cells. Typically, it supports a single flash driver but can be configured to manage multiple devices on request.

> High-performance administration of stored memory data

> Common usage of Fee module by Flash Bootloader (Fbl) and application possible - also with common memory blocks. An update of the ECU software can be done without adjustment of the FBL.
> Redundant storage of management data for increased reliability of data access
> Flexible placement of the Fee sectors in the DataFlash
> Services for handling of undervoltage situations
> Frequently used data is isolated from extremely important data by introducing partitions. This further increases data availability in fault situations (e.g., reset while writing or erasing data).
> Update support for adjustment of nonvolatile memory after ECU re-programming. This is done with a new configuration table (content and size).

12.3 Memory Access (MemAcc)

The MemAcc module provides a memory device agnostic address-based memory access for different upper layers modules using memory drivers (Mem). It implements all high-level functionalities like job management, access coordination, prioritization and allocation of memory driver access requests according to the physical segmentation.
> Access coordination of different upper layers like FEE, EA, OTA software update client etc.
> Memory technology device agnostic API supporting all kinds of memories – data flash, code flash, EEPROM, RAM, PCM etc..
> Virtualization of memory areas to support mapping of non-contiguous areas as well as spanning areas across different memory devices
> Multi-binary synchronization (between different CPU cores) for memory access
> Parallel processing of multiple asynchronous jobs
> Splitting of Memory Driver access requests according to physical segments like pages and sectors for flash memories

MemAcc at Vector is included in selected items, like veOtaDl or the Mem driver.

12.4 Memory Abstration Interface (MemIf)

The MemIf module provides unified access to Ea and Fee services (allows multiple instances of these modules).

12.5Non-Volative RAM Manager (NvM)

The NvM module manages, reads, and writes data to a non-volatile memory (Ea or Fee), synchronizes the data in the RAM areas of the application at system start and shutdown, and saves of redundant blocks for a higher level of data protection. With its NvDataInterfaces, the Rte provides a simple and flexible interface to Nv data.
> Allocates RAM for CRC memory storage
> Dedicated interface for the Dcm diagnostic module for direct read-out and modification of data blocks
> Additional configurable transaction security (standard feature of the Fee module)

12.6 Bulk Non-Volatile Data Manager (BndM)

This module offers an API to read non-volatile data directly from flash memory. BndM is used for bulk data which is frequently read, but not often updated, e.g. certificates. The BndM therefore avoids RAM mirrors.

13. SYS

System-Related Basic Software Modules

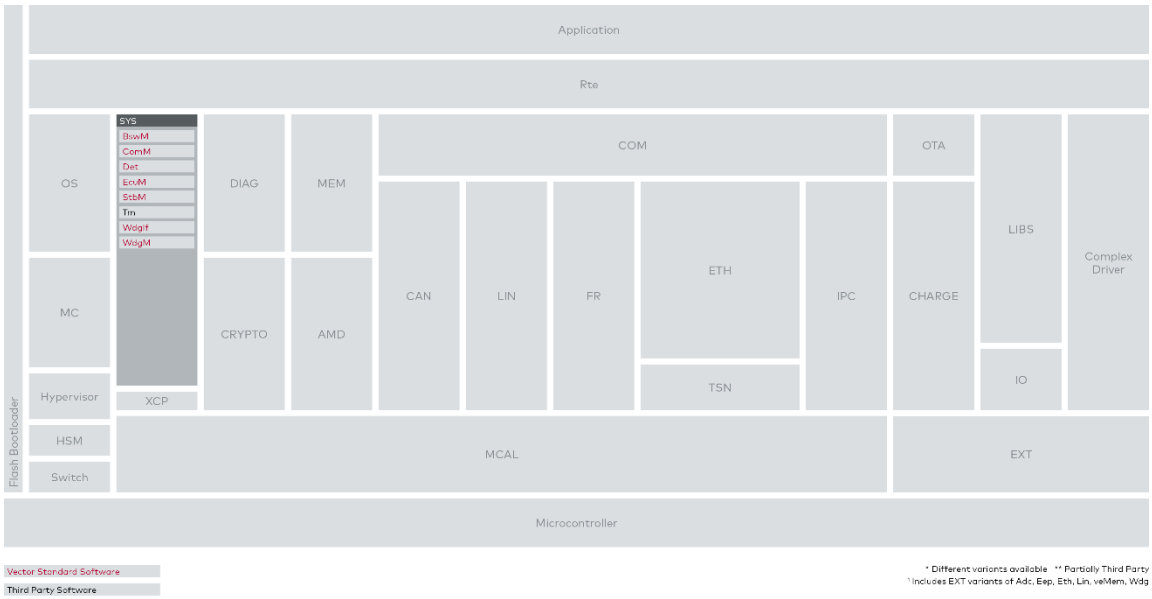

FIGURE 13: SYS cluster

The Basic Software modules in SYS provide essential system services for the ECU. They are accessed by functional software through the Runtime Environment (RTE). These modules include watchdog functions available as Safety Elements out of Context (SEooCs) for safety applications up to ISO 26262/ASIL D. They help in monitoring task runtimes and coordinating software components. The system services manage power, communication channels, component monitoring, and Basic Software module scheduling.

> The EcuM module is included either as a resource-saving precompiled variant or as a flexible post-build solution
> Easy configuration of the initial BswM and EcuM by wizards in DaVinci Configurator Classic
> Creation of initialization sequences
> Configuring a state machine of the ECU like start-up and shutdown
> Management of communication modes

13.1 Basic Software Module Manager (BswM)

The Basic Software Manager (BswM) is essential for handling mode changes within an ECU. It processes requests from software components and modules, executing predefined actions based on user-configured rules. A key function of the BswM is managing the activation and deactivation of PDU groups, which is crucial for diagnostic capabilities. Although its configuration can be complex due to interdependencies, it effectively manages states similarly to a state machine and ensures proper initialization and deactivation of modules.

This process is significantly enhanced by Vector's DaVinci Configurator Classic, which automates complex tasks, making the configuration of Basic Software state handling, module initialization, and PDU group switching more straightforward. The tool offers guidance, parameter validation, and notifications regarding necessary reconfigurations when changes occur, greatly aiding efficient Basic Software setups in automotive projects.

> Support of partial networking
> Support for subnetworks
> Many features for expanded convenience in configuration
> Switching on and off the analysis of rules at runtime
> Realization of a timer (expiration of BswM analyzed), which can be started and stopped by actions
> The BswM can create all necessary mode declarations (if not all necessary SWCs are already available)
> Allows bottom-up configuration

13.2 Communication Manager (ComM)

The Communication Manager monitors the state changes of the communication channels connected to the ECU and of the sub-networks configured in the ECU. It can keep the ECU awake and ready for communication as necessary. Furthermore, it coordinates access of all SWCs to the communication channels and sub-networks. Optionally, ComM supports the bus type “Internal”.

> Support of partial networking
> Compliant with OSEK Nm

13.3 Default Error Tracer (Det)

The Default Error Tracer collects the development errors of the SWCs and Basic Software modules. Optionally, Det supports the Service Ports.

13.4 ECU State Manager (EcuM)

The ECU State Manager deals with startup, shutdown, and wake-up functions within an ECU. Initially, it used predefined operating states, but now it allows flexible state definitions using the Basic Software Manager (BswM). This flexibility supports the implementation of custom energy-saving modes and tailored power-up behaviors.

The module adheres to the AUTOSAR EcuM Flex specification for handling complex state transitions, but it can also operate as per the AUTOSAR EcuM Fixed specification, which includes functionalities like the Run Request Protocol and specific service SWC interfaces. This approach provides a high level of configuration options and supports complex state transitions while remaining compatible with simpler specifications.

13.5 Synchronized Time Base Management (StbM)

The StbM module enables precise time synchronization between different parts of the ECU software. It abstracts the bus-specific time synchronization modules for the CAN, FlexRay and Ethernet by the provision of communication services for vehicle-wide time synchronization.

13.6 Watchdog Interface (Wdglf)

The module Wdglf enables uniform access to services of the Watchdog Driver (Wdg), such as mode switching and triggering. For safety-related ECUs, the Wdglf module must be developed according to ISO 26262.

Add-Ons

> Precise supervision of defined time windows for the watchdog (even for high resolution window watchdogs)

13.7 Watchdog Manager (WdgM)

The module WdgM monitors the reliability and functional safety of the applications in an ECU. This includes monitoring for correct execution of SWCs and Basic Software modules and triggering of the watchdogs at the required time intervals. The WdgM module reacts to potential faulty behavior with multiple escalation stages. An important fact for safety-related functions according to ISO 26262 is the monitoring of correct flow sequences of critical tasks (logical supervision). For safety-related ECUs, the WdgM must be developed according to ISO 26262.

> Precise supervision of defined time windows for the watchdog (even for high resolution window watchdogs)
> The Watchdog Manager monitors the correct operation of the functional software with the modules Wdglf and Wdg (MCAL).

Add-Ons

> Program Flow and Deadline Monitoring: observe the SWCs regarding program flow and deadline monitoring

14. DIAG

Implementation of UDS, OBD and J1939 Diagnostics

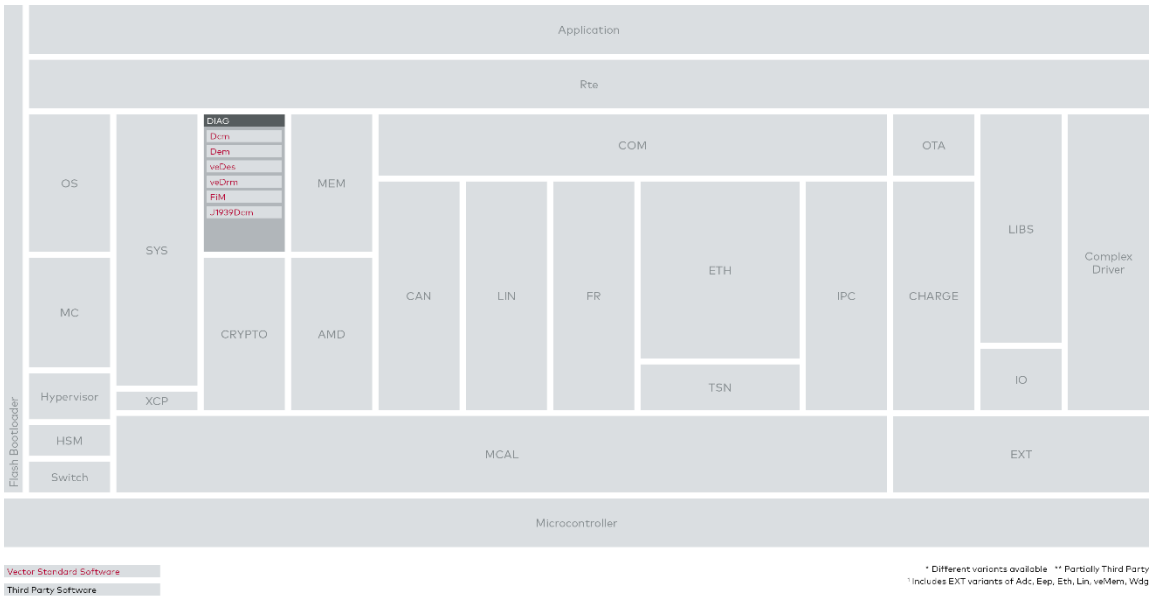

FIGURE 14: DIAG cluster

DIAG includes Basic Software modules engineered for deploying critical diagnostic protocols such as OBD on UDS, OBD-II, WWH-OBD (Euro VI, EURO 7, CN6 HD), ZEV-OBD and J1939 and CN6. This suite is designed to be flexible, accommodating OEM-specific extensions, and is suitable for ECUs even when explicit diagnostic specifications are absent, making it an OEM-independent solution. DIAG is effective for variant diagnostics configuration and optimized for resource usage. In practical applications, DIAG reduces redundancies in ECU software by unifying interfaces, data, services, and diagnostic trouble codes (DTCs). This standardization minimizes overlap and promotes streamlined operations.

For diagnostic data management and development, Vector's CANdelaStudio complements DIAG by providing a comprehensive toolset for authoring diagnostic descriptions, interfacing smoothly with DIAG to support a robust diagnostic solution. This integration ensures compliance with legal standards, such as EURO emissions regulations, and enhances functionality across many OEM implementations.

> Supports OEM-specific implementations while offering OEM-independent capabilities for ECUs without specific diagnostic specifications
> Manages and operates fault memory across multiple cores, including deactivation of functionalities based on active fault entries
> Implements diagnostic protocols like UDS, OBD-II, J1939, and allows for sending and receiving diagnostic messages essential for on-board tester operations
> Configurable using tools like AUTOSAR Diagnostic Extract, CANdela, and formats like ODX
> Supports offline and online calibration with A2L-based common tools
> Optimized for handling more than 30 variant configurations efficiently
> Provides optimized application code templates to streamline implementation

14.1 Diagnostic Communication Manager (Dcm)

The Dcm module implements UDS and OBD-II services in the ECU. An OEM-independent variant of the Dcm is available. The Dcm modules for specific OEMs implement the specifications of the respective OEM. Contact us for detailed information.

> Variant handling for diagnostic configurations

> Easy integration of the Vector Flash Bootloader
> J1939Dcm: Dcm module specially designed for heavy-duty vehicles

Add-Ons
> OBD (including OBD-II, OBDonUDS, WWH-OBD and ZEV-OBD)
> Distributed Diagnostics

14.2 Diagnostic Event Manager (Dem)

The Dem module contains the implementation of the respective OEM requirements concerning the fault memory of an ECU. The OEM-independent variant is available. It supports the following functions as standard features:
> Management of all DTC status bits according to the UDS standard
> Definition of individual snapshots and extended data records
> Predefined extended records (e.g., OccurenceCounter)
> Counter and time-based error de-bounce algorithms
> Displacement of low-priority errors when memory is full
> Flexible unlearning (aging) of errors
> Variants handling for diagnostic configuration
> Link time configuration
> Compressed Configuration Data to optimize code size
> Support of combined errors
> Multi-core support with Dem-Satellite
> Suitable for "mixed AUTOSAR" projects
> Post-build loadable and post-build selectable

Add-Ons
> OBD2: OBD-II (ISO 15031 / SAE J1979) with support for master and primary OBD ECUs
> WWH-OBD: ISO 27145
> HD-OBD
> OBD on UDS
> ZEV-OBD
> HD-CN6
> HD-J1939-OBD
> OBD DTC Split
> BSW Parameter Calibration

14.3 Vector Diagnostic Event Synchronizer (veDes)

veDes enables diagnostic monitoring over multiple MCUs. veDes implicates multi-controller Dem functionality: The diagnostic master collects qualified events communicated from the Dem on the diagnostic slaves, which receive event messages locally.

14.4 Vector Diagnostic Request Manager (veDrm)

veDrm sends diagnostic requests to the same or other ECUs and receives their answers. It provides an API for the application to send/receive UDS services. veDrm allows managing parallel connections, identifying other ECUs in the car and provides a firewall to block potentially dangerous diagnostic requests. The module behaves like an externally connected tester and provides the basis for the implementation of an on-board tester.

14.5 Function Inhibition Manager (FiM)

The FiM module contains the functional features of AUTOSAR 4 as standard.

Add-On
> OBD

14.6 Diagnostic Communication Manager for J1939 (J1939Dcm)

J1939Dcm implements Diagnostic Messages of the SAE J1939-73 protocol (e.g., for reading out the fault memory) in parallel operation with Dcm.

14.7 Asn1Parser

Standalone ASN.1 parser for standards-compliant validation of formats and capable of parsing arbitrary ASN.1 structures. This module can be used as an alternative to the module KeyM.

Main features:
> validate formats such as X.509 and OCSP (with planned long-term support for AC, CVC, and CRL) against the corresponding standards
> parse arbitrary ASN.1 data
> supports lazy parsing, allowing ASN.1 data to be accessed efficiently without parsing the entire ASN.1 structure. This requires that the data format has already been validated in a previous step.

15. MCAL

Drivers for Microcontroller Peripherals

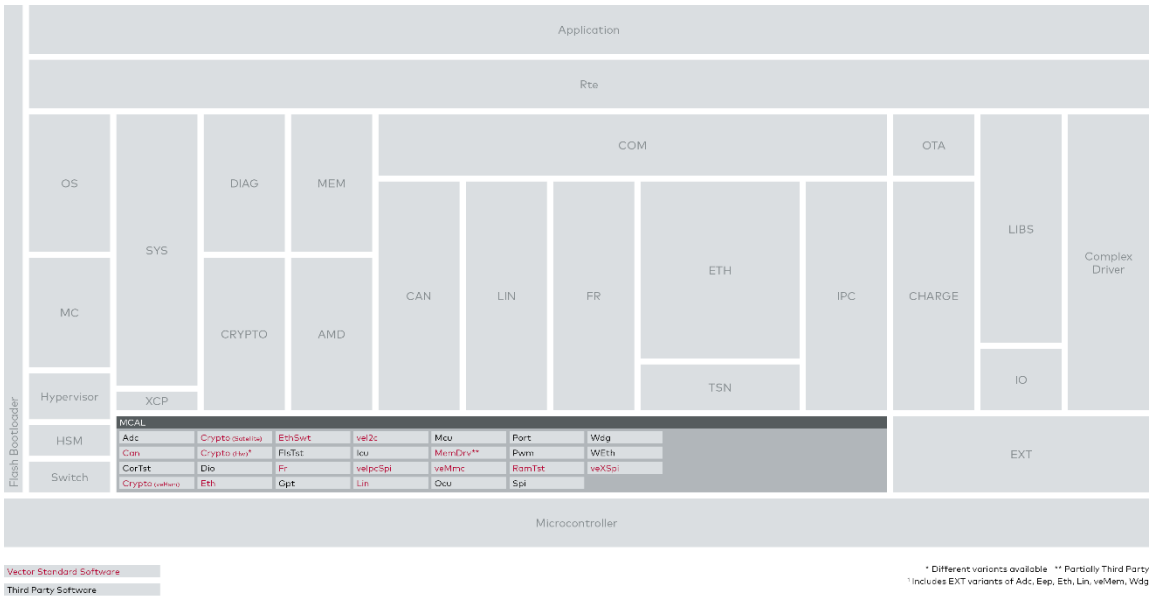

FIGURE 15: MCAL cluster

MCAL contains drivers for controlling the peripherals for almost all microcontrollers and is optimized for specific controllers. The drivers are fully compliant with the AUTOSAR specifications of the Micro-Controller Abstraction Layer (MCAL). When changing the hardware, it is not necessary to make changes to the functional software, only the corresponding driver must be replaced.

> Perfect support of microcontroller peripherals
> Simplified configuration by taking cross-module parameter dependencies into account in the configuration tool
> Development accelerated by plausibility and completeness tests in the configuration tool
> Resource-saving, since functionalities can be deactivated
> Collapsed hardware requirements by optimized utilization of hardware buffers
> Gateway developments are supported by efficient helper functions

15.1 CAN Driver (Can)

The CAN driver abstracts access to the CAN hardware for sending and receiving messages and for switching between controller states (sleep, stop, etc.).

> Notification (Callback) on message reception and after successfully sending a message. This allows to automatically execute application-specific code.

Add-Ons

> HighEndFeature: Offers extended filter options for Multiple Basic CAN objects, Rx queue to shorten the interrupt time during reception, individual polling of mailboxes to assure data consistency and reduction of interrupt load.
> Integrity: Enhances the CAN Driver to work with GHS Integrity OS.
> BSW-Split Multi-Instance CAN: Allows distributed processing of individual CAN channels on different Os cores with the goal of balancing Basic Software workload on several cores

15.2 Can (SocketCAN)

The SocketCan API is a hardware independent API for triggering CAN communication under Linux. The module uses an existing SocketCAN API to realize CAN communication allowing to run a MICROSAR Classic stack under Linux.

15.3 Crypto Driver (Crypto Hw She, Crypto veHsm)

This is the hardware-based abstraction on third-party crypto drivers. The Crypto (Hw) module acts as the driver for accessing security algorithms and functions, which are provided via a Hardware Trust Anchor (HTA). Different HTA types are available such as Secure Hardware Extensions (SHE) and Hardware Security Modules (HSM). Vector offers the following options for the Crypto (Hw) module according to the hardware platform and the derivative used:
> Integration of a Crypto (Hw) developed by Vector
> Integration of a third-party Crypto (Hw) developed at the semiconductor manufacturer

15.4 Crypto (Satellite)

The Crypto (Satellite) driver allows to access crypto accelerator components of MCUs directly by the host cores whereas key provisioning shall be done by the HSM Devices.

15.5 Ethernet Driver (Eth)

The Eth Driver abstracts access to the Ethernet hardware for sending and receiving data and for switching between controller states.

Add-Ons
> Time Sync: Time synchronisation
> TimeSync PhcCorr: Precision Hardware Clock Correction
> Tsn: Time sensitive networking

15.6 Ethernet Switch Driver (EthSwt)

The module EthSwt provides a uniform and hardware independent interface for controlling and configuring Ethernet switches. It also coordinates the MAC learning when using multiple identical ECUs like cameras for the surround view.

Add-Ons
> Time Sync
> Diagnostics
> Tsn

15.7 FlexRay Driver (Fr)

The Fr driver abstracts access to the FlexRay hardware for sending and receiving data and for switching between controller states.
> Supports self-diagnostics. When the FlexRay controller detects an error, it notifies the application so that it can call up the error status
> Optimized wake-up during operation (WUDOP)
> Support of the following APIs: CancelTransmit and L-PdU reconfiguration
> Precompile optimizations, e.g., for single-channel systems

15.8 Vector I2C Driver (vel2c)

The vel2c driver provides services for communication with external I2C chips. vel2c supports master UseCase.
> Contains drivers for interfacing to external peripheral chips via the Inter-Integrated Circuit Bus (I2C)

15.9 LIN Driver (Lin)

The Lin driver provides services for initiating frame transmission (header, response, sleep-mode and wake-up), and for receiving responses, checking the current state, and validating wake-up events.

15.10 Vector Multimedia Card Driver (veMmc)

The veMmc driver is a platform specific communication driver which is used by a veMem (ext) driver to access the external memory.

15.11 Ram Test (RamTst)

The module RamTst tests internal microcontroller RAM cells. A complete test is triggered during start-up and shutdown of the ECU or by a diagnostic command. During normal operation, a periodic test is performed (block-by-block or cell-by-cell).

15.12 Gateway Hardware Routing Manager (veGwHwRM)

Allows to extract routings for offloading to hardware modules or routing engines. Configuration setup of routing modules like controller drivers or Cdds (routing engines).

16. EXT

Drivers for External Devices

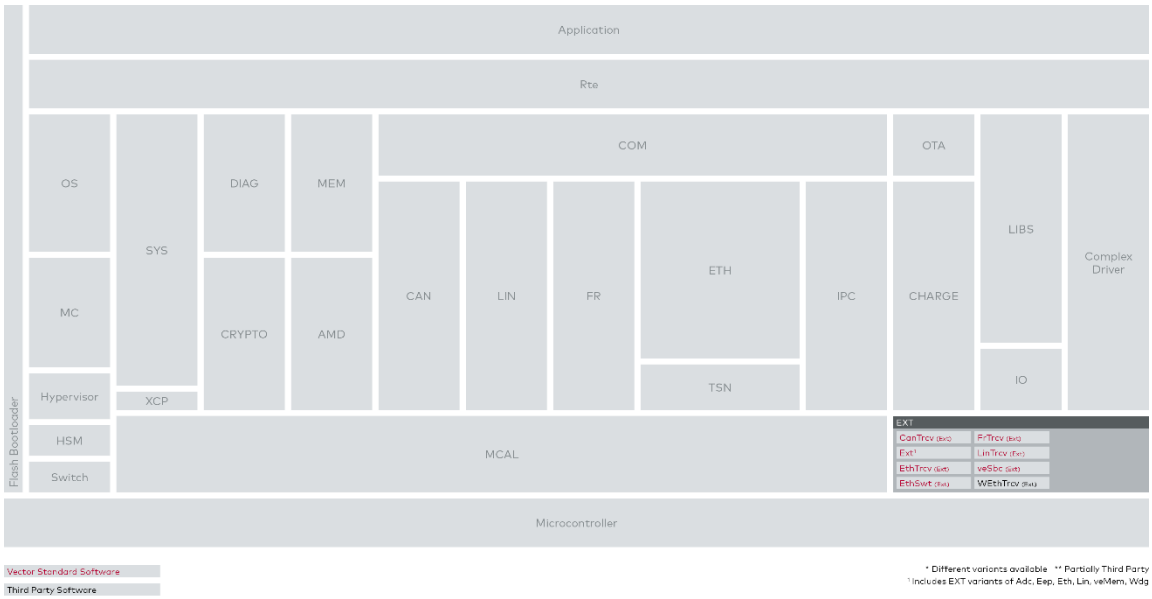

FIGURE 16: EXT cluster

EXT contains communication related AUTOSAR compliant transceiver drivers for CAN, FlexRay, LIN, and Ethernet as well as drivers for other external devices (EEPROM, flash memory, watchdog). The functions they contain are specified by AUTOSAR in the ECU Abstraction Layer, each optimized for a specific device and available for most devices. When changing the hardware, it is not necessary to make changes to the functional software, only the corresponding driver must be replaced.

> Optimal control of external transceivers and memory elements
> Support of transceivers for partial networking and additional support of LIN and Ethernet transceivers
> Simplified configuration by considering parameter dependencies with other modules
> Development acceleration by plausibility- and completeness checks in the configurator
> Partial networking in CAN networks requires special transceivers (CanTrcv can be used for most transceivers)
> Optional services to integrate the configuration of own- or third-party drivers in DaVinci Configurator Classic (seamless and fast configuration of the entire ECU software in one tool)

16.1 External CAN Transceiver Driver (CanTrcv Ext)

The CanTrcv (Ext) module is responsible for controlling the operating states of an external CAN transceiver. It contains control of wake-up and sleep functions. The CanTrcv is also available for Partial Networtking (PN) and for hardware which supports firewall functionality.

16.2 External Driver (Ext)

The Ext module offers drivers for externally connected devices (Adc, Eep, Fls, Eth, Wdg).

16.3 External Ethernet Transceiver Driver (EthTrcv Ext)

The EthTrcv (Ext) module offers a uniform and hardware-independent interface for driving multiple transceivers of the same kind. The Configuration of EthTrcv (Ext) is transceiver-specific and considers the properties of the physical network that is used.

> Also available for MACphy / Powerline Communication (PLC) or Wireless LAN on dedicated hardware.

Add-Ons
> Diagnostics
> MACsec (on dedicated hardware)
> WoDI (on dedicated hardware)

16.4 External Ethernet Switch Driver (EthSwt Ext)

The module EthSwt (Ext) provides a uniform and hardware independent interface for controlling and configuring external Ethernet switches. It also coordinates the MAC learning when using multiple identical ECUs like cameras for the surround view.

Add-Ons
> Time Sync, Diagnostics, MACsec, Tsn, HwPackageFilter

16.5 Host-managed Firewall

This module includes firewall and intrusion detection functionality.

16.6 External FlexRay Transceiver Driver (FrTrcv Ext)

The FrTrcv (Ext) driver for an external FlexRay transceiver is responsible for switching the transceiver on and off.

16.7External LIN Transceiver Driver (LinTrcv Ext)

The LinTrcv (Ext) driver for an external LIN transceiver is responsible for monitoring and driving the wake-up and sleep functions (also for AUTOSAR 3).

16.8 External Abstraction Layer for System Basis Chips (veSbc Ext)

veSbc (Ext) provides an abstraction layer for an externally connected System Basis Chip (SBC). The implementation provides a hardware independent interface that can be used by upper layers to control the SBC hardware. Depending on the SBC peripherals, veSbc (Ext) includes AUTOSAR compliant upper layer modules (e.g., CanTrcv, LinTrcv and Wdg) that allow the basic software stack to access the related SBC functionality.

Further SBC functionalities can be accessed by a complex driver or an I/O hardware abstraction using the RAW-API that is provided by the veSbc (Ext) implementation. The RAW-API allows basic (read/write) access to SBC registers. veSbc (Ext) requires a suitable MCAL driver (e.g., Spi) to establish the connection to the SBC. veSbc (Ext) is also available as a generic version.

17. AMD

Monitoring and Debugging

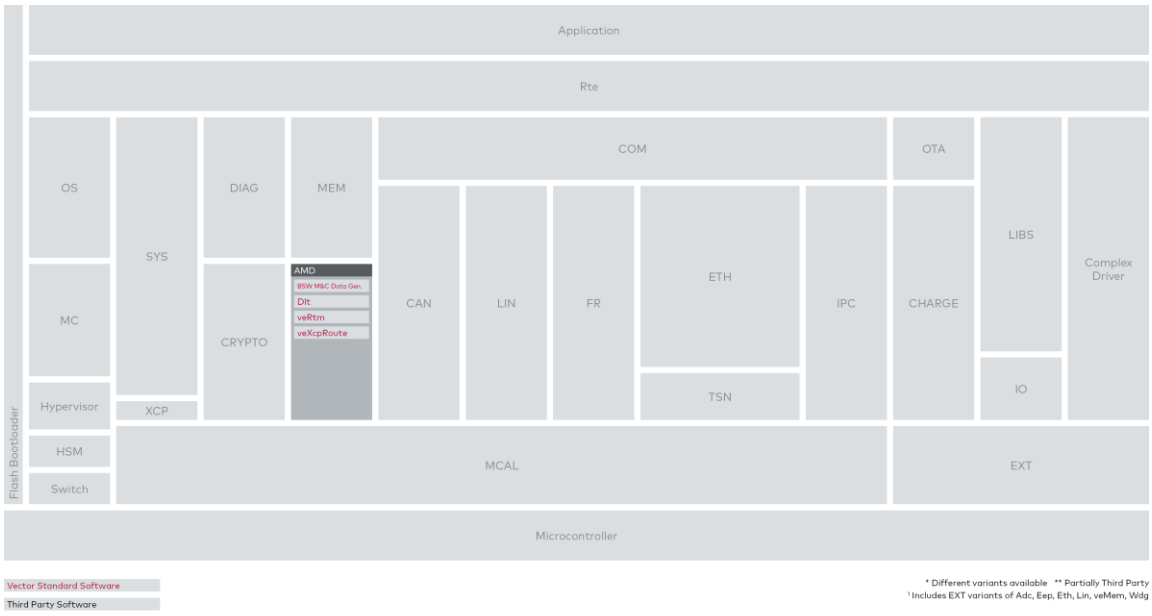

FIGURE 17: AMD cluster

AMD simplifies ECU development, tests, and analysis by passing important information to Vector tools such as CANape or CANoe.AMD via the XCP (Universal Calibration Protocol), which is ideal for passing ECU-internal parameters, in conjunction with a description of the measurement objects (ASAM A2L file). Core functions are the notification of status, errors, and events from the application and the Basic Software, as well as information about the real time CPU utilization and the execution times of the software. Runtime measurements can be performed on Basic Software, applications and CPU. With usage of RTE as runtime environment it is possible to monitor the data flow between SWCs via XCP.

> Easy measurement of start-up and go-to-sleep behavior
> Determination of processor load and execution times of the application and the Basic Software
> Automatic generation of the required A2L file
> Reporting of Basic Software error states and program flow trace notifications
> Usage of the widely used XCP protocol standardized by ASAM
> AUTOSAR 3 support

17.1 BSW M&C Data Generation (subset of former module Dbg)

This module is used in ECU tests and integration for determining internal state of the BSW modules and enables the access to this state variables via XCP.

17.2 Diagnostic Log and Trace (Dlt)

During runtime, the Dlt module records all messages and warnings that the Basic Software reports to the Det module and transmits them via XCP. It also actively forwards all notifications from the fault memory (Dem) to the XCP master. To monitor the program flow and the status of the application, it can report any trace messages in plain text with the help of an API (like WriteLine). Any text or alternatively predefined text blocks optimized for execution time can be transmitted.

17.3 Vector Runtime Messurement (veRtm)

The veRtm module is developed to verify performance goals and to improve code efficiency by profiling - while keeping it easy to use, simple to integrate and universal. veRtm measures user defined code sections (in Applications and Basic Software), response times of asynchronous activities, interrupt locking times, SWC runnables and more. On the CPU it measures code section specifics and the overall CPU load in terms of explicit measurement (time spent in idle tasks) and implicit measurement (interruption times of dedicated background tasks).

17.4 Vector XCP Routing (veXcpRoute)

veXcpRoute uses XCP as standard measurement protocol for vehicle data collection and allows to route XCP messages between different transport layers (e.g., ETH to CAN).

18. IO

Input/Output Hardware Abstraction

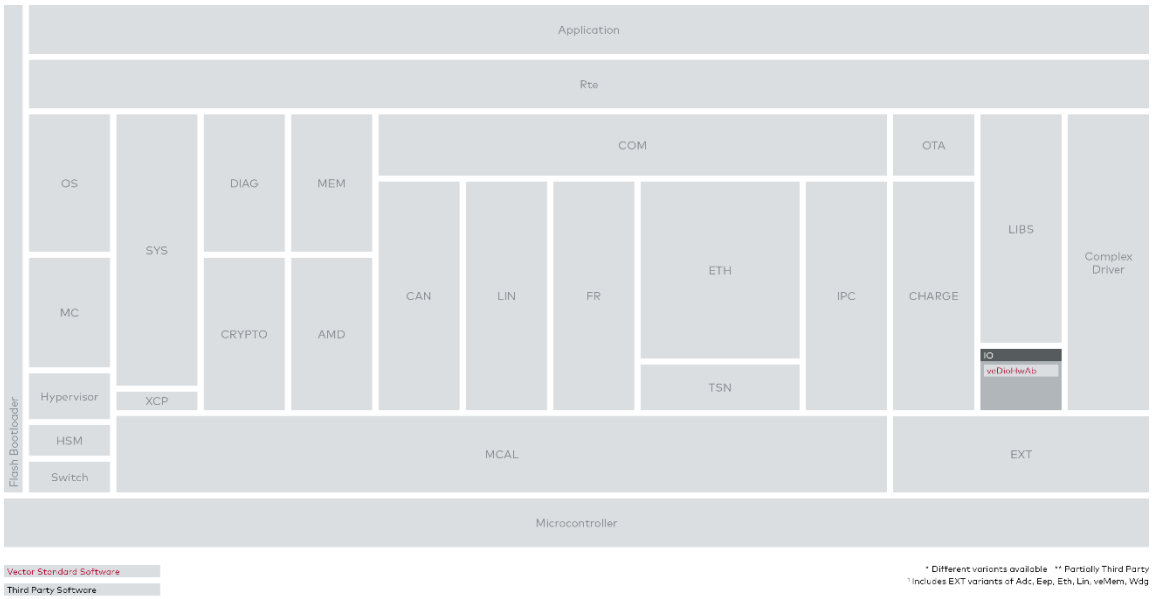

FIGURE 18: IO cluster

IO acts as an interface and establishes the connection between application or SWCs and MCAL modules, allowing access to I/O ports to read sensor data or control actuators.

> Rapid implementation of user-specific code to acquire and provide sensor and actuator signals
> Generation of code examples: User can select read/write templates for digital signals with extensible C code
> Provision of SWC descriptions with all necessary interface definitions
> Matching IoHwAb layer for ECU can be created via DaVinci Developer Classic
> Additional service from Vector to develop complete and ECU-specific IoHwAb layers as part of project work

18.1 Vector Digital Input Output Hardware Abstraction (veDioHwAb)

The Digital Input Output Hardware Abstraction (veDioHwAb) establishes a connection between the application and DIO signals from the MCAL. veDioHwAb therefore creates a SWC interface as well as DIO specific code templates. It offers a quick and easy access to the DIO module. This way, the veDioHwAb module covers a part of the IoHwAb and can be extended by further IoHwAb modules which for example allow access to ADC or ICU channels.

19. RTE

Runtime Environment

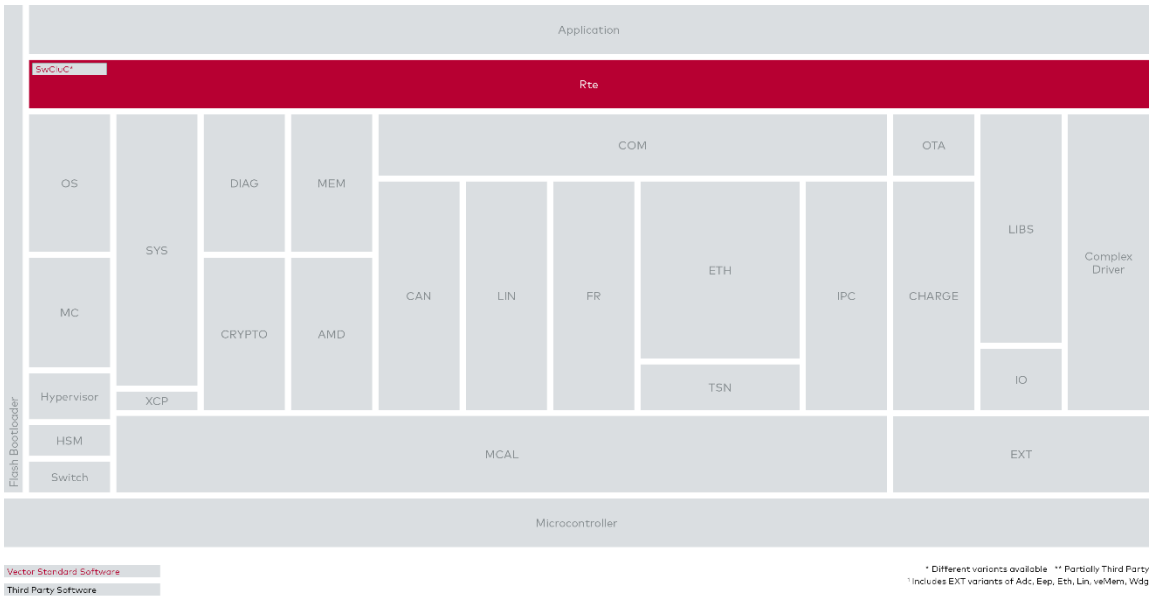

FIGURE 19: RTE cluster

RTE is Vector's scalable and highly optimized AUTOSAR runtime environment. The RTE is a module introduced by AUTOSAR that manages the communication between software components (SWCs), ensures the consistency of the entire information flow and forms the interface between functional software, Basic Software and complex drivers (Cdd). An RTE is required if the functional software consists of AUTOSAR compliant SWCs. In addition to managing the overall event and information flow, the RTE also coordinates accesses across core or memory protection boundaries. ECUs without SWC architecture, and therefore without RTE, are optionally supported by the Bre (Basic Runtime Environment).

> Easy to configure and scalable – modular design, maximum flexibility
> Deep configuration consistency checking
> Highly optimized code with intelligent synchronization mechanisms
> Quickly get started working with AUTOSAR, e.g., based on generated code templates for the software components (SWCs)
> Well suited for migration projects
> Simplified testing of the application through XCP access to S/R ports, InterRunnable variables and per-instance memory
> Reuse in multiple ECU projects of SWCs developed manually or with model-based tools
> RTE only needs to be reconfigured and generated for the specific ECU
> One SWC usable in multiple instances on one ECU
> Contract phase generation for early-stage development of individual SWCs: SWC can be compiled individually (no RTE) and transferred as object code to a development partner
> Generation of the entire RTE: highly efficient and memory-saving. Optimized for the entire ECU configuration, saving resources by keeping execution times short and interrupt disable times minimal (usage of intelligent synchronization mechanisms based on the hardware used)

19.1 Runtime Environment (Rte)

RTE assumed the functionality of the former Schedule Manager/BSW Scheduler (SchM). It defines the tasks and cycle times of the Basic Software modules in the configuration and coordinates the execution of the Basic Software modules. In addition, it defines the settings for the exclusive areas for each module centrally.

> Sender/Receiver and Client/Server communication

> Mode management
> InterRunnable variables, Exclusive Areas, and trigger for runnables
> Access to Nv block software components via sender/receiver ports
> Online and offline calibration of SWCs is supported as well as measurement of S/R ports, InterRunnable variables and Per-Instance memory using the XCP protocol
> Multiple instancing of SWCs and per-instance memory
> Support of the transformer interfaces for ComXf, SomeIpXf and E2eXf
> External Client/Server communication (Inter-ECU) through the optional SomeIpXf.
> Generation of code templates for SWCs based on the “SWC Description”. These templates contain all APIs of the Rte.
> Use of memory protection mechanisms as specified in the AUTOSAR operating system. This support is especially optimized when OS is used.
> Configuration of initialization runnables for the AUTOSAR concept of “Mode-Dependent Runnables”.
> Generation of an A2L file for simple linkage to existing calibration and diagnostic standards.

Add-Ons

> Memory Protection: MPU support for memory protection
> Multi-core support
> Post-build selectable
> Let: Logical Execution Time

19.2 Basic Runtime Environment (Bre)

Compared to the RTE, the simpler Bre is suitable for a less extensive range of functions and accelerates the setup of projects (configurable Basic Software scheduling, management of critical sections, creation of type definitions for Basic Software modules of the service layer). It allows integration of a non-AUTOSAR based application on the Basic Software. A formal SWC design defined by AUTOSAR is not required.

The Bre uses the existing application architecture and provides the foundation when migrating ECUs to an AUTOSAR architecture. The application directly accesses the Basic Software functionality, SWCs do not need to be defined (RTE as middleware between SWCs and Basic Software is replaced by application code). Using the Bre requires the application to adopt the otherwise RTE-typical interfaces. The Bre simplifies the integration of the application with the Basic Software by providing basic RTE functionality.

> Generation of type definitions for service layer Basic Software modules
> Call of main Basic Software functions based on a configurable task mapping
> Management of exclusive areas for Basic Software modules

19.3 Software Cluster Connection (SwCluC)

SwCluC partitions a MICROSAR Classic project into seperately buildable clusters that can be connected to obtain a flashable image. The host cluster contains the Basic Software and optionally SWCs. App clusters contain SWCs, part of the RTE and proxies needed to connect to Basic Software functionality in the host cluster. There is always one host cluster and there can be several app clusters. SwCluC is available in the variants Host and App. SwCluCApp enables cluster providers to manage and develop app clusters. SwCluCHost enables the host cluster provider to develop and manage the host cluster.

19.4 E2E Protection Wrapper (E2ePw)

The E2E Protection Wrapper extends the RTE by adding verification of safety-related signals.

20. OTA

Over-the-Air Software Download

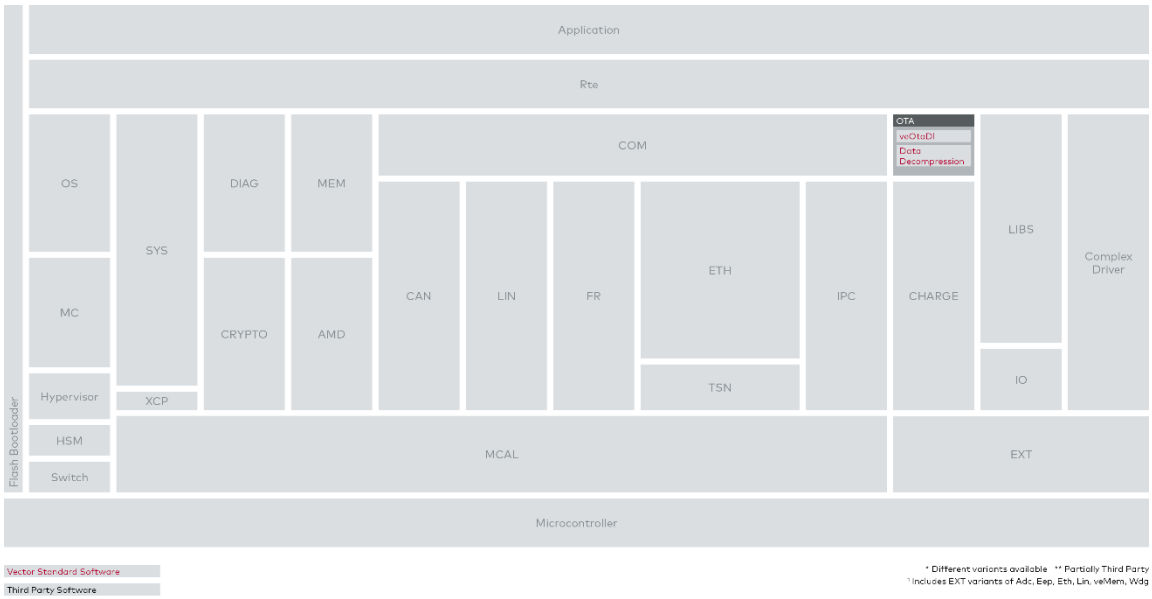

FIGURE 20: OTA cluster

OTA (Over-the-Air) takes care of processing, storing, and activating software updates in the vehicle and provides services to initialize, write and activate them in flash memory. By abstracting the periphery instead of directly affecting physical memory addresses of available internal or external devices, update management is greatly simplified. OEM-specific requirements, which are generally differentiated by their package formats and the update control protocols used, are offered through strategic and flexibly configurable extensions. The download and subsequent update is validated with the help of CRYPTO.

> Efficient data access and configurable management of updateable software areas in nonvolatile memory
> Optimized interaction of MEM and OTA with parallel usage of the ECU's internal memory
> Efficient switching between old and new software by exploiting hardware support (if available)
> Selection of internal or external flashing
> In combination with the Vector Flash Bootloader, the software is updated at the next restart

20.1 Vector OTA Download (veOtaDl)

veOtaDl contains all hardware-independent parts of the software download. Primarily, these are services which are provided for initializing, writing (transferring) and activating software updates. For OEM-specific solutions, veOtaDl offers additional functions.

> Access control during simultaneously use of flash memory by multiple users (MEM and OTA)
> Buffered data handling and routing to the memory driver
> Data compression and data validation
> State management
> Configurable memory abstraction of physical addresses to virtual addresses or software regions

Add-Ons

> Data Decompression LZ77 (Vector): Allows download of compressed data based on a LZ77 compression algorithm (produced by the Vector HexView tool)
> Data Decompression LZMA: Allows download of compressed data based on a LZMA compression algorithm
> Data Decompression LZSS: Allows download of compressed data based on a LZSS compression algorithm

21. LIBS

Libraries

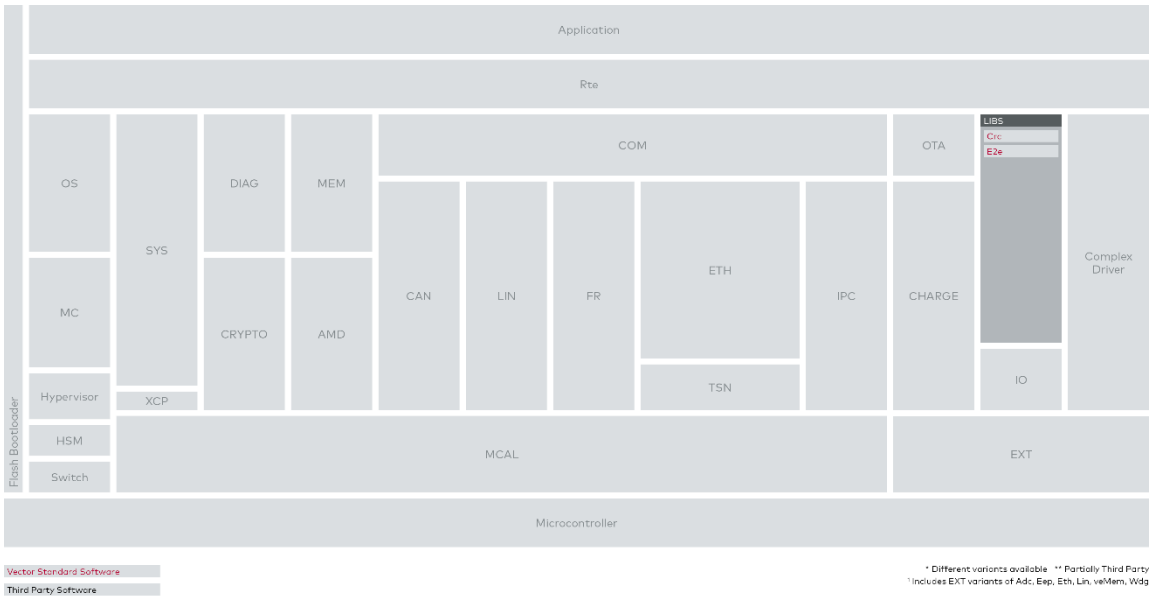

FIGURE 21: LIBS cluster

21.1 CRC

The CRC library contains routines for CRC calculation with methods like table-based calculation, runtime calculation, and hardware supported calculation.

21.2E2E Profiles

E2E protection assumes that safety-related data exchange shall be protected at runtime against the effects of faults within the communication link. By using E2E communication protection mechanisms, the faults in the communication link can be detected and handled at runtime. The E2E Library provides mechanisms for E2E protection, adequate for safety-related communication having requirements up to ASIL D.

22. CRYPTO

Crypto Services

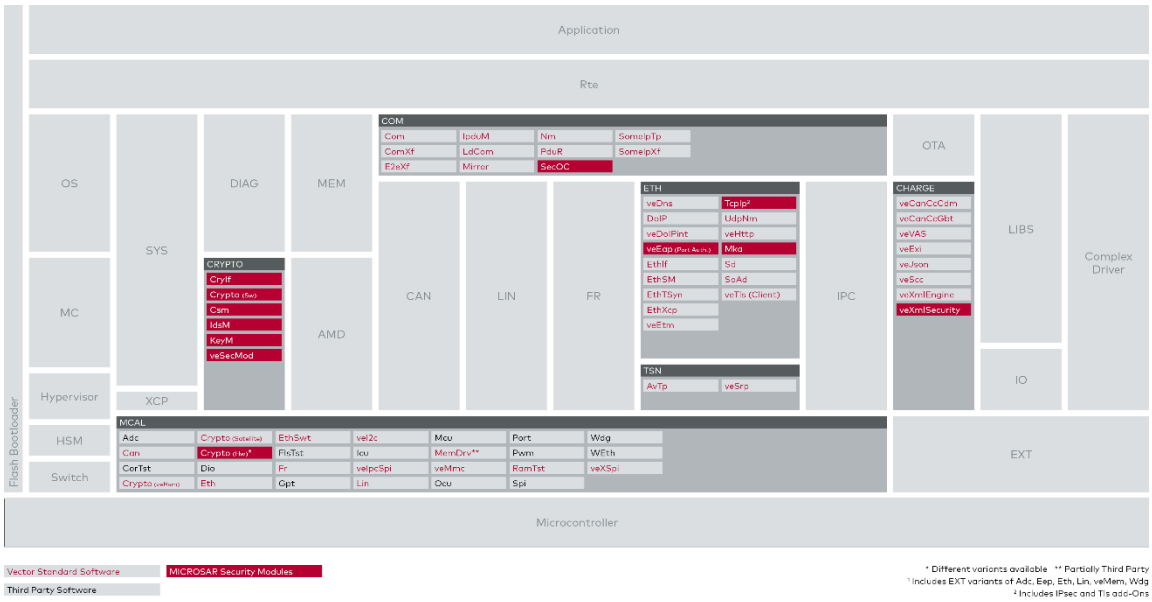

FIGURE 22: CRYPTO cluster

22.1 Crypto Service Manager (Csm)

Csm grants access to cryptographic services and its configuration, also the configuration of algorithms used to execute the services, for synchronous or asynchronous execution, of secure counters and operations on cryptographic keys and certificates.

22.2 Crypto Interface (CryIf)

The Crypto Interface (CryIf) module makes it possible to use CSM hardware-based and software-based crypto solutions. The necessary allocation scheme is managed by the Crypto Interface. The CryIf module provides a uniform interface for different crypto solutions, such as the following:

> Software-based implementation of algorithms which are provided by the Crypto (SW) module
> Hardware-based crypto functions, which are implemented by a Secure Hardware Extension (SHE) or a Hardware Security Module (HSM) via the Crypto (Hw) module.

22.3 Crypto (Sw)

The Crypto (Sw) module provides implementations for cryptographic algorithms and functions in software which are supplied via the CSM. All computations are executed in software, and no special hardware is required to execute cryptographic operations.

Add-Ons

> Symmetric & Asymmetric Algorithms
> PQC Algorithms

22.4 Intrusion Detection System Manager (IdsM)

The Intrusion Detection System Manager (IdsM) receives security events from Basic Software modules, CDD or SWC. Based on its configuration the IdsM filters the incoming security events. Security events which pass the filters are regarded as qualified and can be propagated to the Intrusion Detection System Reporter (IdsR - not included).

22.5 Key Manager (KeyM)

The Key Manager provides capabilities for parsing and verifying certificates based on configurable rules. It uses CSM interfaces for storing certificates and performing cryptographic operations.

22.6 AsnR Parser

The AsnR Parser is a standalone ASN.1 parser for standards-compliant validation of formats and capable of parsing arbitrary ASN.1 structures. The module offers the following main features:

> validate formats such as X.509 and OCSP (with planned long-term support for AC, CVC, and CRL) against the corresponding standards
> parse arbitrary ASN.1 data
> supports lazy parsing, allowing ASN.1 data to be accessed efficiently without parsing the entire ASN.1 structure. This requires that the data format has already been validated in a previous step.

22.7 Vector Security Modules (veSecMod)

The OEM-specific veSecMod includes the Freshness Value Manager (FVM) and the Key Manager (KeyM). FVM provides a freshness value to the Secure Onboard Communication (SecOC) to prevent replay attacks and therefore requires while KeyM handles key exchange and key updates.

MORE INFORMATION
Visit our website for:

> News
> Products
> Demo software
> Support
> Training and workshops
> Contact addresses

vector.com