<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6.1</td>
<td>Overview of Advantages</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>6.2</td>
<td>Application Areas</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>6.3</td>
<td>Functions</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>6.4</td>
<td>Configuration</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>6.5</td>
<td>Handling of Variants</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>6.6</td>
<td>Scope of Delivery</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>6.7</td>
<td>Additional Services</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>7.1</td>
<td>Overview of Advantages</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>7.2</td>
<td>Application Areas</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>7.3</td>
<td>Functions</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>7.4</td>
<td>Configuration</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>8.1</td>
<td>Overview of Advantages</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>8.2</td>
<td>Use Cases</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>9.1</td>
<td>Features and Advantages</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>9.2</td>
<td>Functions</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>9.3</td>
<td>Application Areas</td>
<td>24</td>
</tr>
</tbody>
</table>
1 Solutions with CANbedded

The software implemented in automotive ECUs can be either infrastructure software or application software. The latter implements the desired functionality, while the infrastructure software includes the operating system, communication software and fundamental input and output services.

The use of uniform communication protocols is absolutely necessary when setting up networks in a vehicle. The use of standardized software components ensures easy portability and reliable integration of ECUs in the vehicle's network for the automotive OEM and its suppliers. In addition, the software components lead to a substantial reduction in development costs.

1.1 CANbedded

Vector develops high-quality and innovative software components for use in the automotive industry. One focus of the product line is on ECU communication, an area in which Vector has well-founded knowledge. The necessary hardware drivers are also important here.

CANbedded products from Vector are available for systems without AUTOSAR. You can obtain them for the CAN and LIN networks as well as for the J1939 protocol. In addition, Vector can offer individual solutions for further simplification of the application, e.g. solutions for gateways or state management.

1.2 Advantages of CANbedded

- Many years of experience: Vector has been developing basic software for CAN communication in automotive applications since 1992
- High level of maturity: It would be hard to find a vehicle with a CAN bus today that did not have a CANbedded stack
- Broad range of applications: Applicable for various use cases – from powertrain via body bus to infotainment
- Support of many automotive manufacturers: Available in many OEM-specific variants

1.3 Your Partner Right From the Start

Vector offers comprehensive support, from the start of a project to its successful completion. Training sessions and courses provide basic information and in-depth knowledge on working with the software – conducted in Vector’s classrooms or right at your company. Vector can assist you through the ECU development process with on-site startups, code reviews, project-specific services and other development-supporting software tools. For more information on individual training events and a schedule go to www.vector-academy.de.

This support is supplemented by telephone and e-mail hotlines, special workshops with a focus on specific OEMs and individualized consulting on software integration. This reduces your development time and development costs.

1.4 Uniform Solution

Network communication is described in a database. It is used to configure the software components and serves as a foundation for Vector development and test tools. This gives software developers a universal and uniform solution, from simulation and design to testing and validation.

Further information on tools for development and testing of automotive networks can be found

- For the DaVinci Configurator Pro: please see the separate Product Information
- For GENy: please refer to the section at the end of this document.

2 CANbedded - Basic Software for CAN Communication

Vector CANbedded basic software lets ECUs exchange information over the CAN bus. As a part of the ECU software, it handles communication-related tasks as specified by the OEM. With CANbedded, your ECU is able to efficiently communicate with other ECUs in the vehicle and with an external service tester. CANbedded is available in variants for most OEMs and for a large number of microcontrollers.

CANbedded consists of the configurable software components CAN driver, Interaction Layer, Network Management, Transport Protocol and Communication Control Layer, which you can combine as necessary to form a communication stack. This provides you with a reliable foundation for your ECU.
Figure 1: Overview of the CANbedded Software Components

2.1 Overview of Advantages:

- Reliable and tested through many years of development
- Generated source code that is scalable and application-specific
- Low memory requirement and fast runtimes
- Supports over 80 different hardware platforms
- Available in platform-specific and OEM-specific variants
- Clearly defined and uniform interfaces
- Easy to port to other platforms and OEMs
- Easy to integrate in your ECU software

2.2 Application Areas

CANbedded lets users focus their efforts completely on their functional software, because the CANbedded stack handles OEM-conformant sending and receiving of messages over CAN. Vector offers OEM-specific CANbedded variants for most OEMs. The CAN driver (the only hardware-specific component) is available in an optimized version for many different hardware platforms (8 to 64 bit controllers).

The use of CANbedded in all of a networks ECUs assures compatibility of all communication stacks. This reduces development costs and test effort.

A single-channel variant of CANbedded has been optimized for use in ECUs connected to a single CAN bus. A multichannel variant is also available for communication over multiple CAN channels.

You can extend the CANbedded stack with additional basic software from Vector for your specific application:

- CANbedded Gateway
- Diagnostics: CANdesc
- Commercial vehicles: CANbedded J1939
- Measurement and calibration: CCP or XCP

2.3 Functions

CANbedded provides simple interfaces (APIs) that are identical for all applications. If possible, the API is standardized over all OEM and hardware variants. You can integrate CANbedded quickly and conveniently in the ECU software. The CANbedded components route asynchronous events such as receiving of CAN messages, bus wake-up or different error states (Assertions) to the application by means of configurable callback functions. You can configure the precise set of
generated APIs and callbacks to ideally adapt them to the needs of your specific application. The Vector tool chain optimally supports you here.

CANbedded consists of the software components described in the sections that follow. Vector would be glad to assist you in selecting the components you need.

### 2.3.1 CAN Driver

The CAN driver provides an interface for higher software layers that is as hardware independent as possible. This enables platform-independent use or re-use of the higher software layers.

All necessary settings such as parameters for the hardware acceptance filter or Bus Timing Register are made at configuration time. The configuration tool GENy from Vector offers preselections for configuring these filters and registers.

The following basic functions are provided by the CAN driver:

- Initialization of the CAN controller
- Sending of CAN messages
- Receiving of CAN messages
- Overrun and error handling (Bus-off)
- Notification via wake-up events

### 2.3.2 Interaction Layer (IL)

The IL is available in OEM-specific variants and operates signal-oriented. It is responsible for:

- Sending messages depending on the send types of the signals they contain (Cyclic, OnEvent, IfActive, etc.)
- Timeout monitoring of Rx messages and signals
- Notification of sending and receiving of signals

### 2.3.3 Network Management (NM)

The main task of Network Management is to manage the sleep and wake-up states of a network’s ECUs. This functionality is handled by the NM CANbedded software component, which is available in different variants. Depending on OEM requirements, either OSEK NM, AUTOSAR NM or another OEM-specific NM algorithm is used.

### 2.3.4 Transport Protocol (TP)

By default, the CANbedded Transport Protocol conforms to ISO 15765-2. OEM-specific TP variants are also available, e.g., VWTP or MCNet. The ISO 15765-2 conformant variant is available in either the 1999 variant or the 2004 variant and can be delivered with different addressing methods (Normal, Extended, Normal fixed, Mixed).

### 2.3.5 Communication Control Layer (CCL)

The CCL simplifies integration of the CANbedded software components in your ECU software, such as CAN driver, IL, TP, NM, and other CANbedded options Gateway, J1939 and CANdesc. It manages defined system states (PowerOn, StartUp, ShutDown, Stop/Sleep, Normal) as well as communication states (Net request, Net release). The CCL offers the following functions:

- Initializing the CANbedded software components
- Coordination of state transitions (WakeUp and GotoSleep)
- Calling periodic functions of the CANbedded software components such as Timer Tasks or State Tasks.
- Controlling the bus transceiver

### 2.3.6 Special CCL Functions

- The CCL already contains OEM-specific communication requirements such as Wakeup Prevention and Minimum Communication Activity.
- The CCL supports what are known as multipurpose ECUs For details see chapter "Identity Manager".
2.4 Configuration

You configure CANbedded with the PC-based configuration tool GENy. GENy utilizes information from the communication database (DBC file) as well as manual settings. GENy generates parameterization files based on the configuration. They contain the parameters and function calls of the CANbedded software components that are used.

2.5 Scope of Delivery

You get the CANbedded components as C source code. The scope of delivery of CANbedded includes:

- GENy configuration tool as executable Windows program
- C-header and source files
- Documentation

2.6 Additional Basic Software from Vector That is Relevant to CANbedded:

- CANbedded J1939
- CANbedded Gateway
- CANdesc for diagnostics
Measurement and calibration with XCP/CCP

CANbedded LIN

2.7 Availability

CANbedded basic software is available for many commonly used microcontrollers and in OEM-specific variants. For more information go to: [www.vector.com/ds_canbedded.availability_en](http://www.vector.com/ds_canbedded.availability_en).

2.8 Additional Services

You can have Vector perform project work on the following aspects:

- Adaptation of the CANbedded software components to special target platforms
- Extension of the CANbedded software components
- Creation of software components for other bus systems such as FlexRay or Ethernet based on AUTOSAR

3 CANbedded J1939 - Embedded Software Components for J1939 Applications

The CANbedded J1939 bundle contains standard software components for your SAE J1939-based applications. CANbedded J1939 provides functions for fundamental communication between ECUs in heavy-duty vehicles and agricultural machines. The software components are available as optimized single or multi-channel variants. You can extend CANbedded J1939 with various software components of the CANbedded product line from Vector.

3.1 Overview of Advantages

- Convenient configuration via GUI
- Code and runtime efficiency due to wide-ranging configurability
- Good portability by hardware-independent API
- Easy to integrate in your application software, since it is delivered as source code
- Easy integration of other standard components (e.g. transport protocol, diagnostics, etc.)

3.2 Application Areas

- ECUs of the drivetrain and chassis of heavy-duty vehicles
- FMS (Fleet Management System)
- Terminals and implements for agricultural machines

3.3 Functions

CANbedded J1939 contains a J1939 communication stack with the following software components:

- J1939 Base Module incl. Dynamic Interaction Layer: Contains queues for sending and receiving J1939 messages (PGNs), static address claiming (J1939 NM) and an interface to AUTOSAR-OS and other operating systems. The IL provides standard access to the signals and parameters of PGNs with variable CAN identifiers (node address, priority). Optionally, the J1939 Base Module offers the ability to implement multiple ECUs with different ECU addresses on a single hardware unit (virtual ECU).
- CAN Driver: The CAN driver handles sending and receiving of J1939 messages. It is hardware-dependent and is available for many currently used microcontrollers Details can be found in the chapter “CANbedded”.

As options, you can extend the communication stack with the following components:

- J1939 TP BAM: Transport protocol for broadcast messages per J1939-21
> J1939 TP CMDT: Transport protocol for point-to-point transmission per J1939-21
> J1939 Dynamic NM: Support for dynamic address assignment per J1939-81
> J1939 Router: Routing of J1939 messages to other J1939 buses using configurable filter rules
> J1939 ISOBUS extension: ETP and FastPacket transport protocols, working sets

![Figure 4: CANbedded J1939 Software Architecture](image)

### 3.4 Configuration

You can conveniently configure and generate CANbedded J1939 standard software components with Vector’s PC-based software tool GENy.

![Figure 5: Configuration of J1939 Software Component is Performed with GENy](image)

### 3.5 Scope of Delivery

The CANbedded J1939 product contains the following:

> GENy for configuring and generating software components
> Software components as C source code
3.6 Supported Hardware Platforms

CANbedded J1939 is available for many different controllers (min. 16-Bit) and compilers. You will find a current list online at: www.vector.com/ds_canbedded.availability_en or on request.

3.7 Other Relevant Products

- CANdesc: Embedded software component for OEM-specific diagnostics over KWP2000 according to ISO 14230 or UDS according to ISO 14229. Details can be found in chapter "CANdesc and DEM".
- Transport protocol (TP): Software component for transmitting data > 8 bytes according to ISO 15765-2. In addition, OEM-specific variants are available. Details can be found in chapter "CANbedded".
- MICROSAR OS: Preemptive real-time operating system per the AUTOSAR standard with a resource-efficient, robust operating system core.

3.8 Additional Services

Vector can perform project work for you in the following areas:

- Modification of software components for other target platforms
- Extensions of CANbedded J1939 for ISOBUS (ISO 11783) and NMEA2000
- Extension of standard components by adding customer-specific functions
- Flash programming

4 CANbedded LIN Communication - Embedded Software Components for Local Interconnect Network

LIN (Local Interconnect Network) is a cost-effective, serial communication system for distributed electronic control units in the motor vehicle. The LIN protocol was specified by the LIN Consortium, of which Vector was an Associate Member and is now published in the ISO 17987:2016. Earlier versions of LIN Standard (1.3 and 2.x) are still strongly represented in the market.

Figure 6: Overview of the Software Components of CANbedded LIN Communication
Today, AUTOSAR-based LIN Master nodes are often combined with CANbedded LIN based slave nodes as both products are fully compatible.

4.1 Features and Advantages

Increasing software complexity makes it advantageous to implement standardized software components. Vector’s expertise acquired in developing and working with other protocols has been utilized throughout the development of LIN software components:

- Standard software components based on a portion of the generated code are scalable for application-specific requirements
- Automatic generation of parameter settings and configurations
- Runtime efficiency
- Low demand for ROM and RAM memory
- Simple interfacing to the application software
- Compatibility with other Vector products including analysis, simulation, and calibration tools
- Support of LIN 1.2/1.3/2.0/2.1 and 2.2 as well as J2602 and ISO 17987:2016
- Support of multi-channel LIN configurations and multiple ECU instances (like door left/right) besides many other optional features
- LIN Slave node option for functional safety according to ISO 26262 ASIL A/B

LIN networks are described in the LDF (LIN Description File) format according to the “LIN Configuration Language Specification”. This includes all of the information needed to define signals, messages, baud rate and schedule tables. A configuration tool is used to adapt all LIN software components to ECU-specific requirements. The tool imports either LDFs or NCF files (Node Capability File) for parameterization. Modifications and settings in detail:

- Adaptation of ECU-specific parameters to the network design
- Configuration of the LIN driver to the properties of the specific ECU e.g. oscillator frequency, UART, etc.
- Configuration of diagnostic services for LIN Slave ECUs
- Definition of message and signal callbacks

The source codes of the LIN software components are compiled and linked with the application. Vector’s LIN simulation and analysis tools (e.g. CANoe.LIN and CANalyzer.LIN as well as the SCOPE option) are available for integration, analysis and for test purposes. For physical bus access one of Vector’s VN16xx-Interface is available; serving as universal serial bus interfaces for the LIN bus.

Timing control of the LIN software components is achieved by having tasks called periodically by the application or an operating system. Therefore, the LIN driver does not require any auxiliary timers.

4.2 Functions

The fundamental objective of the implementation is to provide an interface that is easy to use and satisfies all applications. The standardized interface offers these features:

- Initialization of the LIN hardware
- User-configurable baudrate (Recommended: 19200, 10417 and 9600 Baud)
- Sending of LIN messages (confirmation by flags and callback functions)
- Receiving of LIN messages (notification by flags and callback functions)
- Sleep and wakeup handling
- Protocol error handling
- Timeout monitoring of messages
- Checking of runtime behavior during the development phase (debugging)
4.3 Application Areas

LIN was developed to round out a collection of automotive protocol standards, but it has been adopted in other areas such as automation engineering as well. The focus is on cost-effective communication networks for ECU subsystems with low data rates. LIN therefore supplements existing bus protocols.

4.4 Hardware Interfaces

Two software components are needed for LIN communication. One component is a hardware-dependent low-level SIO driver, whose task is to implement abstraction of access to the serial interface. The second component, the LIN protocol driver, is overlaid on this and provides an API to higher-level components. This API satisfies all of the requirements of the current LIN standard.

The driver is a standardized and easy-to-configure software component for communication between master and slave network nodes.

The LIN driver is a software component for communication between master and slave network nodes which is standardized and is easy to configure with a configuration tool. It is one of the CANbedded components from Vector; which assures consistent integration with CAN components. In addition, it may be combined with MICROSAR software.

As a result, this makes it easy to implement CAN-LIN gateways, which are usually realized as master network nodes.

4.5 LIN Master Transport Layer

Since the master ECU usually has an auxiliary CAN channel, over which diagnostics are performed from a tester, the main task of the LIN master is to pass the received tester requests to the relevant LIN slaves. The LIN Diagnostic Transport Layer (DTL) is used for this; it offers two API types according to the LIN 2.x and ISO 17987:2016 specifications:

> DTL Raw is used if the data already exist in segmented form (routing of CAN TP messages on LIN).

> DTL Cooked accepts assembled diagnostic data and segments them according to LIN bus requirements (Diagnostic Service Data Units).

4.6 LIN Slave Diagnostics

LINdiag is a very compact implementation of UDS 14229.1 diagnostics for LIN slave ECUs. It receives requests, routes them to the appropriate service and constructs the response in interaction with the application. This involves internal management of a diagnostic buffer to monitor for data consistency and overruns. To send and receive requests and responses LINdiag needs the LIN Transport Protocol, which is included in the delivery.
4.7 Further Options

- Optional for slave ECUs: Diagnostic software components with associated transport protocol
- Optional for master ECUs: Transport protocol with raw or cooked API
- Gateway: CAN-LIN and LIN-LIN routing for master ECUs
- SAE J2602: As a supplemental option, support of the SAE J2602 “Recommended Practice” specification is available for LIN drivers.
- Flashing of LIN slaves
- Calibration over the XCPOnLIN protocol

4.8 Scope of Delivery

The following items are supplied with the product:

- Configuration tool (executable Windows program)
- C sources for LIN driver and optional software components
- Documentation and operating instructions

4.9 Availability

Our LIN software components for motor vehicle ECUs are available for a large number of commonly used microcontrollers. For further information please visit: [http://www.vector.com/vi_canbedded_lin_en.html](http://www.vector.com/vi_canbedded_lin_en.html).
4.10 Licenses
When the same LIN hardware interface and same compiler are used the acquired licenses can be used for all devices of a processor type and any desired projects or OEMs.

4.11 Supplemental Services
Vector also offers related project services such as:

- LIN driver expansions
- Support of other target hardware
- Development of higher OEM-specific protocol layers, e.g. diagnostic protocol

5 CANbedded Gateway - Standard Embedded Software Components for CAN and LIN Routing
To exchange data between different networks (CAN-CAN, CAN-LIN), a gateway ECU is needed that only transports to the other network those messages or signals that are actually needed.

5.1 Overview of Advantages

- Selectable coupling of CAN and/or LIN networks with user-definable routing relationships
- Mature standard software component
- Solid, lean and efficient foundation for your gateway ECU
- Available for a large number of OEMs

5.2 Application Areas
CANbedded Gateway is a mature standard software component for gateway ECUs between different CAN or LIN networks. CANbedded Gateway lets users define routing relationships simply and quickly. Both 1:1 and 1:N routings are supported in any direction. The CANbedded Gateway utilizes bus-specific standard components of CANbedded, and so it is hardware independent. It covers the specific requirements of many different OEMs.

5.3 Functions
The CANbedded Gateway contains complex mechanisms for receiving all messages and signals and then sending them to another network under consideration of filter and conversion rules. Users can select from:

- Signal routing
- Message routing (with/without data queue)
- TP (Transport Protocol) routing
5.3.1 Signal Routing
Routing of signals makes it possible to exchange dedicated information (e.g. momentary vehicle speed) between the networks. The send cycle times, send types and signal positions within messages – all defined in the communication database – may differ between the source and target networks. The values of the signals to be transmitted are not changed in the process. In case of an Rx timeout, the CANbedded Gateway implements a substitute value strategy. The signal to be transmitted is written and routed with a predefined signal value (default value or special substitute value). If routing relationships occur that cannot be resolved, e.g. if the signals of a routing relationship have different sizes, the application can use C callback functions to flexibly adapt signal contents before routing (e.g. by scaling signal values).

5.3.2 Message Routing
In routing of messages, the entire message is routed in either the interrupt or task context. This routing type is faster and more efficient than signal routing, but it cannot be applied in all routing relationships. In message routing, the messages may differ in their message-ID, send cycle and send type.

By default, message routing always utilizes the most recent data. If data is transmitted less frequently than the data is processed by the associated Rx event, the CANbedded Gateway sends the data of the last Rx event. Neither the signal layout within the message nor the signal values are changed here.

The user can control gateway activities by configuring "conditional routing". This saves on bandwidth during critical phases, such as during software flashing or diagnostic sessions. This is done by defining routing groups and assigning routing relationships to one or more routing groups. At runtime, the application can enable or disable routing of individual groups with the help of an API.

5.3.3 TP Routing
The TP Gateway is responsible for routing the transport protocol data. Diagnostic requests from CAN to LIN and the associated responses from LIN to CAN might be transmitted, for example. Using the ISO transport protocol component from CANbedded, it is possible to support different protocol addressing modes on the CAN bus.

5.4 Other Relevant CANbedded Products
Preconditions for use of the CANbedded Gateway are
> when signal routing is used: the presence of the Vector Interaction Layer
> when TP routing is used: the ISO Transport Protocol for CAN or the DTL (Diagnostic Transport Layer) for LIN
> and the relevant Vector CAN or LIN drivers. The "extended functions CAN driver" option is required (Low Level Message Transmit) for the CAN driver.

![Figure 9: Layer Model of the CANbedded Gateway](image-url)
5.5 Configuration
The CANbedded Gateway is configured with GENy, the configuration and code generator tool from Vector. The configuration process is based on a DBC or LDF/NCF description file. Routing relationships are defined either
- Automatically by OEM-specific algorithms or
- Manually by targeted selection of the signals and messages to be implemented.

5.6 Scope of Delivery
- Configuration tool as executable Windows program
- C header files and C source code
- Documentation

6 CANdesc und DEM - Software Components for Diagnostics

6.1 Overview of Advantages
- CANdesc covers important OEM-specific requirements
- Quick, simple and uniform interface to the diagnostic protocol in your application
- Together with CANdelaStudio: Comprehensive diagnostic solution from specification to embedded software
- Guaranteed consistency between diagnostic specification and generated code

6.2 Application Areas
The diagnostic software component CANdesc (desc = diagnostic embedded software component) is the uniform implementation of the diagnostic protocol for different automotive OEMs. CANdesc is generated according to the configuration based on the diagnostic specification (CDD file). To implement the error memory, you can extend CANdesc with the DEM component, which gives you a fully diagnostic solution for your ECU.

CANdesc and DEM are options of the CANbedded communication stack from Vector. However, you can also easily use the two software components in other communication stacks for e.g. FlexRay or IP networks.

6.3 Functions
CANdesc and DEM are available for a number of different OEMs. An OEM independent version is also available. CANbedded is available for UDS, CANdesc supports both KWP2000 and UDS. The components contain the following functionalities:

6.3.1 CANdesc
- Provides a signal-based interface as a uniform interface to the application. This simplifies interfacing of diagnostics to your ECU software.
- Full implementation of diagnostic services $28 (CommunicationControl), $2A (ReadDataByPeriodicIdentifier) and $2C (DynamicallyDefineDataIdentifier)
- Validation of diagnostic requests in reference to Service ID, Session, Sub-Service ID, Format, Session and Security level of a sub-service
- State control (depends on the CANdela database, e.g. Session or Security level)
- Assurance of data consistency in parallel diagnostic requests from the same tester or multiple testers
- Generation of a directly compilable template including the necessary function stubs that lets you realize your own diagnostic implementation quickly and easily.

6.3.2 DEM
As soon as the application software detects errors, it must initiate saving of the related Diagnostic Trouble Codes (DTC). The DEM component can be used for this task. For each error (DTC), the software component stores the following information in error memory:
DTC status byte
> Environment data
> Cycle counters
> Fault detection counter
> Aging counter

When DEM is combined with the diagnostic software component CANdesc, you get a complete implementation of the following error memory related UDS services:

> $14 Clear Diagnostic Information
> $19 Read DTC Information
> $85 Control DTC Setting

Figure 10: Complete Generation of the ECU’s Diagnostic Software

6.4 Configuration

You configure CANdesc with the GENy tool and a CANdela file (CDD). To create this file, you will need the tool CANdelaStudio Version 6.5 or higher. For configuring DEM, the DaVinci Configurator Pro is needed.

The use of CANdelaStudio offers CANdesc users significant advantages. For more details, please refer to the CANdelaStudio Product Information.

6.5 Handling of Variants

If your ECU requires diagnostic configuration variants, CANdesc and DEM offer a powerful solution for this. With GENy you can define up to 31 different parameter sets and store them in the ECU in a resource optimizing way. This avoids redundancies in the ECU software, because identical interfaces to the same data, services or DTCs are merged in the generated diagnostic code.

6.6 Scope of Delivery

> GENy configuration tool as executable Windows program
> C source code for DEM; CANdesc is fully generated as C code using GENy
> Documentation

6.7 Additional Services

Vector offers project services related to diagnostics:

> Adapting the application to CANdesc
> Implementing error memory strategies
> Process consultation in the area of diagnostics
> Testing diagnostic implementations for ECUs

Figure 11: Diagnostic Software (CANdesc and DEM) in the Diagnostic Development Process

7 XCP Professional - Measuring, Calibrating, and Testing

7.1 Overview of Advantages
> Can be activated/deactivated for production operation
> Easy to configure with PC-based tool
> Low memory requirement and short execution times based on scaling of the software component to your requirements
> Transport layers available for CAN, FlexRay and Ethernet
> Compatible with MICROSAR BSW modules (AUTOSAR)
7.2 Application Areas

Automotive in-vehicle ECUs are measured, calibrated, and tested with the XCP Protocol (Universal Calibration Protocol) – the successor to CCP (CAN Calibration Protocol). XCP was standardized by ASAM (Association for Standardization of Automation and Measuring Systems), a process in which Vector played a leading role. By clearly differentiating the Protocol Layer and Transport Layer, XCP is able to support many different bus systems.

On the PC side, the CANape or CANoe tool is used as the XCP master. Its counterpart in the ECU is the XCP slave. This is implemented by the XCP Professional software component. It not only contains the ASAM functional content, but useful extensions as well.

You can use XCP Professional together either with a conventional communication stack or in AUTOSAR systems. In either case, it is unnecessary to modify the XCP software. If a software stack by a third-party producer is used, small modifications to the XCP Transport Layer (TL) may be necessary. Vector would be glad to assist you in this area.

7.3 Functions

XCP Professional contains all necessary functions for communicating with the XCP Master, such as:

- Synchronous data acquisition based on DAQ lists
- Read and write accesses to memory addresses
- Initialization and switchover of the memory area for calibration data
- Support of time stamps
- Protection against unauthorized writing and reading in memory
- Access protection by Seed & Key support
- Synchronous data stimulation (STIM) and Resume mode
- Flash and EEPROM programming
- Communication mode: Block transfer
- Transmission of service request packets
- Bit modification and short download
- Generation of an A2L file based on the ECU configuration

7.3.1 Synchronous Data Stimulation (STIM)
To optimize algorithms and parameters, you can use XCP over the bus system to read measured signals from the ECU (DAQ) and modify it with a simulation model (e.g., MATLAB). The re-computed values are then written back to the ECU (STIM). Reading and writing are performed synchronously in a predefined time interval (Bypassing).

### 7.3.2 Resume Mode

The Resume mode permits cold-start measurements, e.g., to check ECU behavior in the wakeup phase. This involves activating the XCP stored in the flash memory immediately after a cold-start – even before the application starts. Sending of the measured values begins automatically, and they can be logged for later evaluation.

### 7.3.3 Flash and EEPROM Programming

The Vector XCP component is not only able to store measured values in RAM, but also in flash or EEPROM memory. To do this, XCP executes callback functions in which the application could activate the flash algorithms, for example. This gives you the option of individually adapting the algorithms to the flash or EEPROM memory being used.

### 7.3.4 Communication Mode: Block Transfer

Block Transfer lets you transfer large quantities of data. This functionality is comparable to that of a transport protocol. The sender segments the data, and the receiver re-assembles them in the correct sequence.

### 7.3.5 Transmission of Service Request Packets

Service request packets let you transmit lines of text. They are stored in program code, and when the program section is executed they are sent to the XCP master where they are displayed. This function is very useful primarily during system development and in debugging.

### 7.4 Configuration

XCP Professional is configured with the PC-based GENy configuration tool. Compiler switches may be used to deactivate unneeded functions, and this reduces memory loading. When XCP transmission is started, the list of supported functions is communicated to the XCP master via the XCP plug-and-play mechanism.

### 7.4.1 XCP-Master

The XCP slave provides the XCP Master with access to measurement signals and calibration parameters in the ECU with the help of an ECU description file in ASAM format (A2L).

### 7.4.2 CANape as XCP Master

CANape is the all-round tool from Vector for measurement, calibration, and diagnostics of ECUs. The key properties of CANape related to measurement and calibration are:

- Time-synchronous real-time acquisition and visualization of internal ECU signals with XCP/CCP, of signals from CAN, LIN, Ethernet and FlexRay buses and from external measurement equipment
- Online calibration via XCP/CCP, real-time stimulation via XCP and offline calibration
- Flash programming of the ECU via XCP/CCP or the diagnostic protocol
- Rapid prototyping for functional development by bypassing a MATLAB/Simulink model with the help of CANape

Additional information can be found in the separate Product Information on CANape.

### 7.4.3 CANoe.XCP or CANoe.AMD as XCP-Master

CANoe is Vector’s tool for the development of both complete ECU networks and individual ECUs. CANoe is an XCP Master with its XCP and AMD options.

CANoe.XCP provides you with reading or writing access to the ECU memory location via the ASAM-standardized XCP or CCP protocol. Specifically for AUTOSAR, CANoe.AMD (AUTOSAR Measurement and Debugging) contains the “State Tracker” window that displays discrete states and binary signals in a user-friendly way. This allows you to analyze critical flow sequences.

For more information, please refer to the separate Product Information on CANoe.
Identity Manager - Solution for Multiple ECUs

Identities are used to identify ECUs. It determines which variant of the ECU should be activated (e.g. which door side). An identity is the configuration of an ECU that is active at runtime, and it is based on a description file. The identity is set when the ECU is initialized.

When tracking the development of ECUs in automobiles, one trend is clearly evident – ECUs are becoming increasingly more powerful. At the same time, the number of variants is growing drastically, which is reflected in the growing effort required to manage them and also results in growing costs.

Cost savings are realized by reducing administrative effort. This can only be achieved with the assistance of intelligent solutions that can manage the many different variants.

Based on our many years of working together with automotive OEMs and suppliers, Vector knows these continually recurring use cases well and has developed optimized solutions especially for them, to simplify the development of ECUs, reduce administrative effort and finally to thereby save on costs.

8.1 Overview of Advantages

- More efficient handling of variants
- Less administrative effort
- Reduced warehouse costs

8.2 Use Cases

Practice shows that it is primarily the following three use cases that recur frequently.

- Physical multiple ECU
- Virtual multiple ECU
- Multiple configurations ECU

8.2.1 Physical Multiple ECU

A typical case of physical multiple ECUs are the ECUs for the driver and passenger side doors. These ECUs have nearly identical functions. However, there are differences in the receive and send messages, diagnostics and Network Management address.

The steps: Develop an ECU that will later be used for both door sides. Load the database that covers the ECUs in the GENy configuration tool and select both ECUs simultaneously. Configure the Identity Manager as a physical multiple ECU and assign a separate identity to each of the nodes (Left ECU, Right ECU).

There are essentially two different ways to proceed in coding: If the message layout is identical between the identities, utilize the manual buffer overlay in GENy. This means that the same signal APIs can be used regardless of the currently active identity. So, only one application needs to be developed, but the ECU acts differently to the outside world depending on the identity which is initialized. The differences are automatically read out from the DBC file.

In the second case, buffer overlaying is not needed. Instead, differentiate between the left and right sides using generated macros in the code. It is also easy to implement crossovers of receive and send messages between the two door ECUs while optimizing resources.
This software is loaded in the left and right ECUs. The decision on whether the software is to be installed in the left or right door ECU is made with each new initialization. A simple function call establishes the ECU's identity.

### 8.2.2 Virtual Multiple ECU

![Image of Virtual Multiple ECU]

Higher-performance control modules also contain more memory space. This makes it possible to represent multiple ECUs (e.g. gateway and radio) in one CPU, saving on additional hardware. These are referred to as virtual multiple ECUs.

The different communication nodes are derived from one database. In the configuration tool GENy, you only need to select those nodes that will be implemented and then configure a virtual multiple ECU in the Identity Manager. All nodes are assigned to one identity. The identity operates from a node superset and all necessary information on the nodes.

### 8.2.3 Multiple Configurations

![Image of Multiple Configuration]

A typical application involving multiple configurations is the development of ECUs for both a current car line and a new car line. These ECUs are also referred to as carry over ECUs. Changes made within the associated database are often very extensive. It is possible to use a completely different signal layout or use different messages for different model series. The new configuration could even support a different number of CAN buses.

First, develop the software for both car lines. In doing so, treat each car line as though it were assigned to one (or more) dedicated CAN channel(s) with an associated database (DBC file). These configured, logical CAN channels are then mapped to physical CAN Controllers. Just as with the physical multiple ECU, the ECU’s identity is determined by a function call at the start.

### 9 GENy - PC-Tool for Configuring Embedded Software Components

GENy is the convenient configuration tool for software components from Vector. It is used to configure CANbedded components and the Flash Bootloader for use in production ECUs.
9.1 Features and Advantages

- Thoroughly modular structure due to its static framework, stand-alone graphic user interface (GUI) and dynamically reloadable components
- Reloadable hierarchies make it well suited for any configuration task (CANbedded, LIN)
- A cleverly designed framework mechanism interprets and manages dependencies in configuring components
- XML is used throughout as the file format, both for the component descriptions (Attributes) and data saving
- Open interfaces offer simple expandability

This makes GENy a scalable and generic configuration editor/generator that permits consistent configuration from a single integration environment, even in complex heterogeneous systems.

Vector’s many years of experience in the area of ECU development ensures optimal workflows between OEM and suppliers without altering their tasks or roles.

GENy is a collection of components (not a rigid program block); therefore it can also be expanded very easily by adding auxiliary components. GENy’s graphic interface makes it very easy to use and intuitive. GENy always appears with the same “look and feel” independent of the components to be configured. The configuration data is shown and processed in a hierarchical tree structure. This makes navigation user friendly, and the user gets a clear view of all options.

The Configuration View can be modified manually or via an external XML file. The latter is essential for an OEM-specific appearance of GENy’s GUI and guarantees that parameters will have functionally relevant preset values.

Tool tips and the online help function help to save time and immediately gives you the information needed about possible component settings. Plausibility checks prevent invalid settings and thereby prevent errors.

The generation process is logged in the Message Window, and this indicates potential errors (e.g. mistakes in configuring) and other important information. This is done to avoid tedious debugging efforts.

Characteristics:

- Configuration templates for different application types
- System-wide parameters only need to be configured once
- There is only one XML-based configuration file
- Expandable due to open interfaces
- For Windows XP and Windows 7 (32 bit and 64 bit)
9.2 Functions

GENy is the tool for configuring CANbedded software components. Configuration and parameter files can be generated to integrate them in the application project. Functional capabilities of GENy include the configuration of:

- Embedded software components for CAN, LIN
- Flash Bootloaders
- AUTOSAR BSW (communication modules)

GENy reads the following input files:

- CAN database files (DBC)
- LIN files (LDF)
- FlexRay files (Fibex)
- CANdela database (CDD)
- AUTOSAR “ECU Configuration Description” (EDUC)

9.3 Application Areas

The GENy configuration tool is always used in conjunction with other Vector software components for communication and diagnostics. GENy fulfills AUTOSAR Consortium requirements related to the AR3 standard.