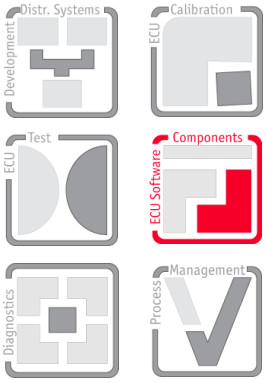


User Manual

Startup with PSA

A Step by Step Introduction

Version 1.0.0

English

vector

Manual History

Author	Date	Version	Details
Klaus Emmert Manuela Scheufele	2009-07-24	0.1	Creation
Klaus Emmert Manuela Scheufele	2009-10-01	1.0.0	Released

Reference Documents

No.	Source	Title	Version

Impressum

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Inhaltsverzeichnis

1 Manual Information

In this chapter you find the following information:

1.1 About this user manual

Finding information quickly

The user manual provides the following access help:

- At the beginning of each chapter you will find a summary of the contents,
- In the header you can see in which chapter and paragraph you are,
- In the footer you can see to which version the user manual replies,
- At the end of the user manual you will find an index, with whose help you will quickly find information,
- Also at the end of the user manual you will find a glossary in which you can look up an explanation of used technical terms

Conventions

In the two following charts you will find the conventions used in the user manual regarding utilized spellings and symbols.

Style	Utilization
bold	Blocks, surface elements, window- and dialog names of the software. Accentuation of warnings and advices. **[OK]** Push buttons in brackets **File|Save** Notation for menus and menu entries
MICROSAR	Legally protected proper names and side notes.
`Source Code`	File name and source code.
Hyperlink	Hyperlinks and references.
<CTRL>+<S>	Notation for shortcuts.

Symbol	Utilization
	Here you can obtain supplemental information.
	This symbol calls your attention to warnings.
	Here you can find additional information.
	Here is an example that has been prepared for you.
	Step-by-step instructions provide assistance at these points.
	Instructions on editing files are found at these points.
	This symbol warns you not to edit the specified file.

1.1.1 Certification

Certified Quality Management System

Vector Informatik GmbH has ISO 9001:2000 certification. The ISO standard is a globally recognized standard.

Spice Level 3

The Embedded Software Components business area at Vector Informatik GmbH achieved process maturity level 3 during a HIS-conformant assessment.

1.1.2 Warranty

Restriction of warranty

We reserve the right to change the contents of the documentation and the software without notice. Vector Informatik GmbH assumes no liability for correct contents or damages which are resulted from the usage of the documentation. We are grateful for references to mistakes or for suggestions for improvement to be able to offer you even more efficient products in the future.

1.1.3 Registered trademarks

Registered trademarks

All trademarks mentioned in this documentation and if necessary third party registered are absolutely subject to the conditions of each valid label right and the rights of particular registered proprietor. All trademarks, trade names or company names are or can be trademarks or registered trademarks of their particular proprietors. All rights which are not expressly allowed are reserved. If an explicit label of trademarks, which are used in this documentation, fails, should not mean that a name is free of third party rights.

➔ Outlook, Windows, Windows XP, Windows 2000, Windows NT, Visual Studio are trademarks of the Microsoft Corporation.

1.1.4 Errata Sheet of manufacturers

Caution: Vector only delivers software!

Your hardware manufacturer will provide you with the necessary errata sheets concerning your used hardware. In case of errata dealing with CAN please provide us the relevant erratas and we will figure out whether this hardware problem is already known to us or whether to get a possible workaround.

Info: Because of many NDAs with different hardware manufacturers or because we are not informed about, we are not able to provide you with information concerning hardware errata of the hardware manufacturers.

2 Getting Started

In this chapter you find the following information:

2.1 How to use this Manual

Step by Step	Just follow the description step by step.
Basic Information	To find basic information about CANbedded (see section Basic Information on page 29).
FAQ	To find answers to special questions without reading the whole document use the FAQ list (see section FAQs on page 45).

2.2 Start immediately or need Basic Information?

You are Novice or Expert?

This User Manual is designed to fit the needs and expectations of the developers of the ECUs. Of course there are differences in planning the software architecture. But the core is almost the same for all types of ECUs.

Your aim is to implement the CANbedded software components as fast as possible. Perhaps you already know the basic concepts of CANbedded?

Then let's start with the step-by-step introduction in how to startup with PSA CANbedded software components regardless of the ECU type. You will find remarks if the handling differs for a specific ECU type.

For more basic information about CANbedded refer to (see section Basic Information on page 29).

3 A few STEPS to Basic ECU with PSA CANbedded

In this chapter you find the following information:

3.1 STEP What do you need before start?

CANbedded

Did you get the CANbedded delivery?

3.2 STEP Installation

The following list shows the tools and software (C Code or Library) that are included in the CANbedded delivery and what has to be installed further on.

- **Vector CANbedded SIP CBDxxxxxxx Rxx <name>_Setup.exe**
  BSW modules, exact content depends on your delivery, the details are outlined in the following illustration.
- **GENyFramework_<version>-PGP-sda.exe**
  downloaded from the FTP server. The framework of the configuration tool GENy.

Your Delivery from Vector

Unpack the Delivery

Start the **Setup.exe** and follow the installation dialogs.

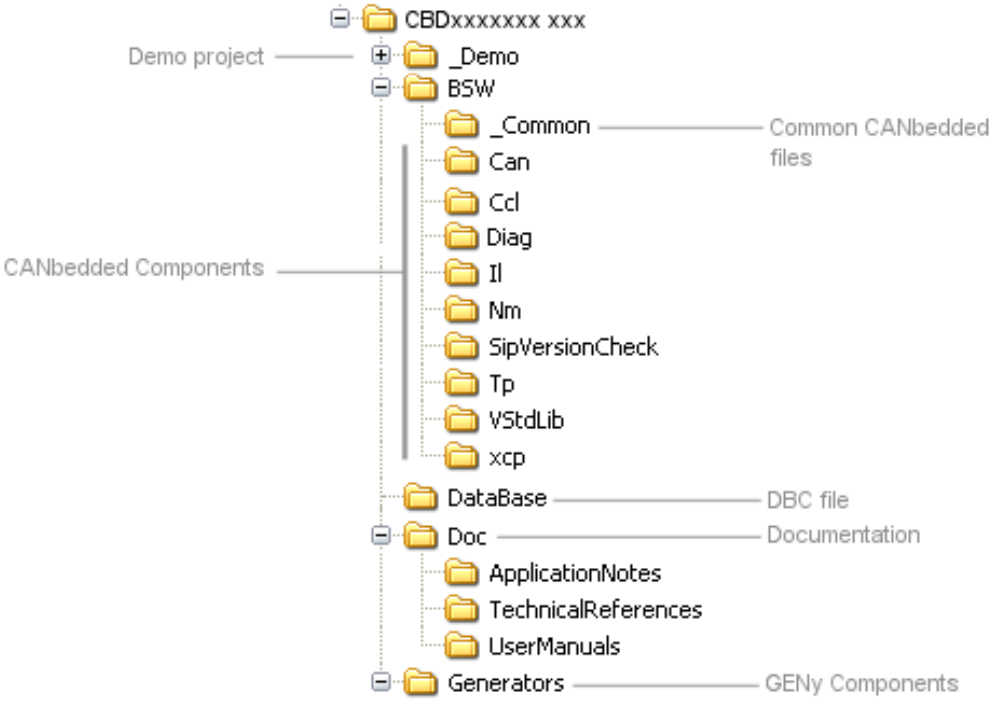
Situation after installation

Use the **Start|Programme|Vector CANbedded …|** to find the installation.

Try to keep a proper file structure to keep the overall view throughout the complete development process

You will find the software components in the following file structure or in a similar one.

Example File Structure after Installation



Info: It is up to you to use a different file structure. This is merely a recommendation and the result of the installation process.

3.2.1 BSW folder

You will find the following files in the BSW folder:

CAN Driver

CAN - CAN Driver

can_drv.c – can_def.h – can_inc.h – cancel_in_hw_user_cfg.cfg

(delete underscore for usage)

Info: Dependent on the CAN Driver there could be additional files.

Network Management

NM – Network Management

Generic_precopy.c – INM_Osek.c – INM_Osek.h – Stat_Mgr.c – Stat_Mgr.h

Interaction Layer

IL - Interaction Layer

il.c – il_def.h – il_inc.h

Transport Protocol

TP – ISO Transport Protocol

tpmc.c – tpmc.h

Diagnostics Layer

Diag - Diagnostics Layer CANdesc

The files will be generated completely

Communication Control Layer

CCL - Communication Control Layer

ccl.c – ccl.h – ccl_inc.h

_Common

v_def.h – v_ver.h – sip_vers.c – sip_vers.h – vstdlib.c – vstdlib.h

Info: The SIP check ensures that all used CANbedded components fit together. If not, a pre-processor error will occur. Please make sure that the SIP check file is complied with your application.

Universal Measurement and Calibration Protocol

XCP - Universal Measurement and Calibration Protocol

_xcp_appl.c - _xcp_appl.c - xcp_can.h - XcpProf.c - XcpProf.h

3.2.2 Installation GENy Framework

Install GENy Framework

Open the link from the delivery description to open the FTP server and save **GENyFramework_<version>-PGP-sda.exe** on your computer. Start **GENyFramework_<version>-PGP-sda.exe** and enter the passphrase (password from e-mail) for GENy Framework.

The GENyFramework_<version>.exe will be unpacked. Start this setup file, follow the installation hints and enter the path to your delivery. GENy.exe will be installed at this path.

3.3 STEP Configuration Tool and DBC File

Follow the explanations for the basic settings of the CANbedded software components in the GENy configuration tool.

3.3.1 Start GENy with a Link or a Batch File

Working with GENy

There are different ways to start GENy and to load a configuration. By default you can start the configuration tool and load the configuration afterwards. The more comfortable way of using GENy is via a link or a batch file where you can start the tool directly within your project with your project as a parameter.

Start GENy with a Link or a Batch File

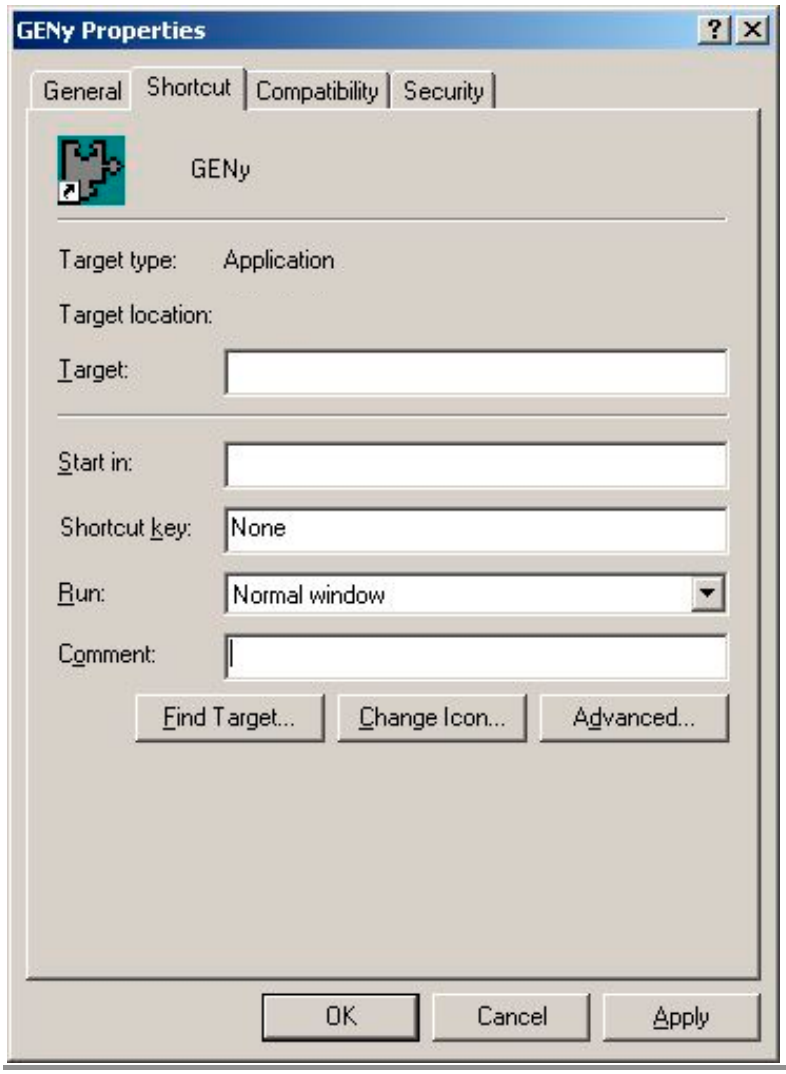
This is the recommended way if more than one CANbedded stack or configuration setup is to be used in parallel. Link and batch file must contain the following information:

- Path to the **GENy.exe**
- Path to component **DLLs** and **license** file
- Optional: Path to an existing configuration file **.GNY**.

Example: Link

Enter the following path information into the Target area of the MS Window control:

yourPath\GENy\geny.exe /m ..\yourPath_GENy\Components /c ..\yourPath\GENyProject.gny



Example: **Batch file**

Please write all commands below in ONE line into the batch file!

..\yourPath\GENy\geny.exe /m ..\yourPath_GENy\Components '/c ..\yourPath\GENyProject.gny

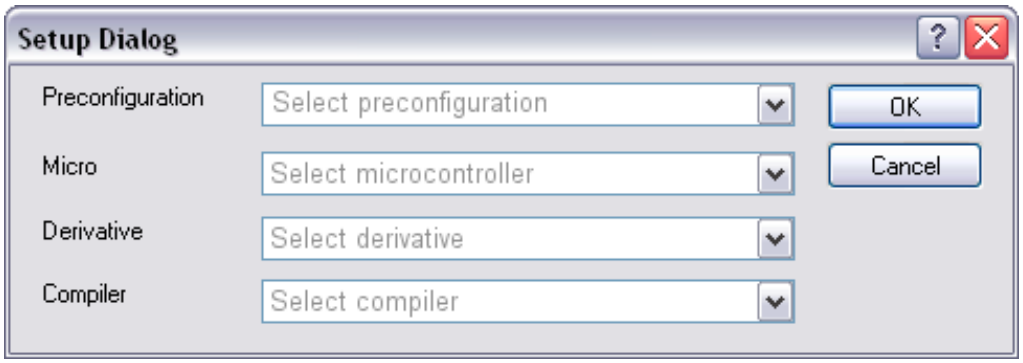
3.3.2 Preparations in GENy

New Configuration

Create a new configuration via File / New or the New Button .

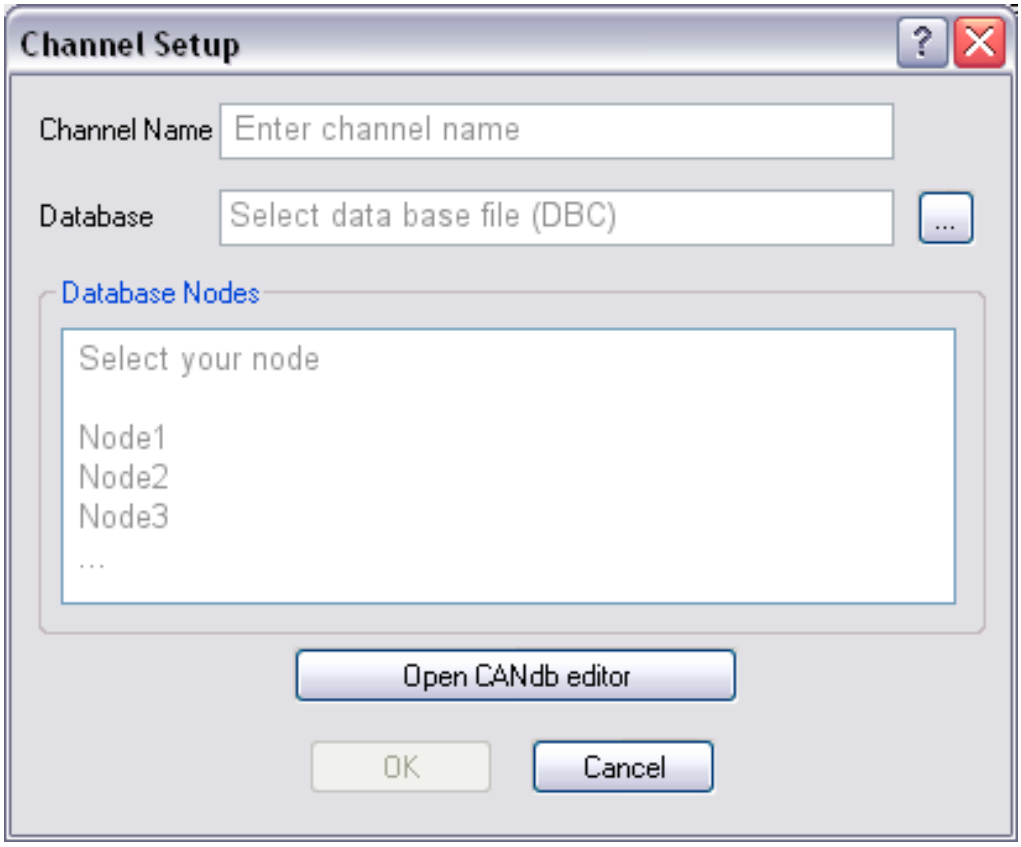
Setup Dialog

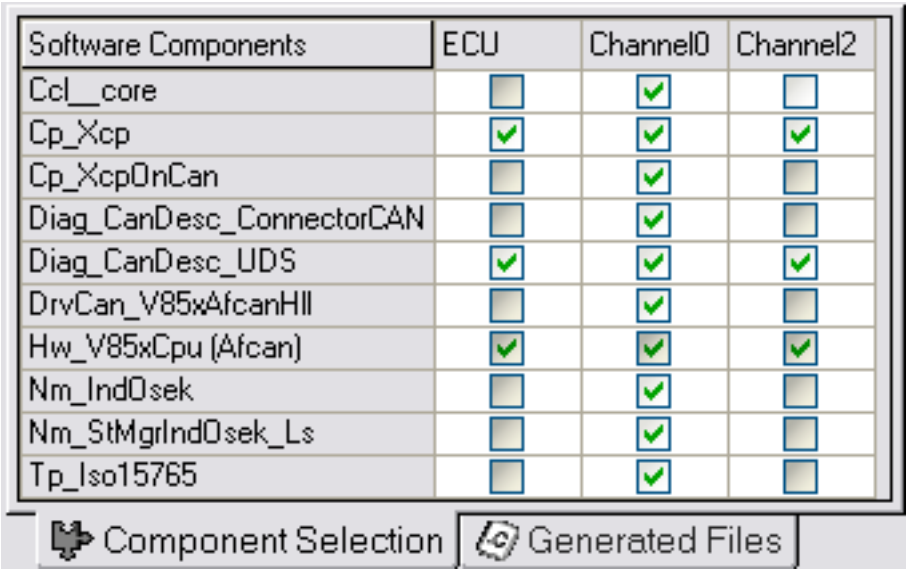
Select your **Preconfiguration**, **Micro**controller, **Derivative** and your **Compiler** in the setup dialog.



Data base file and Component Selection

Click on  to add a channel and select CAN as channel type. In the Channel Setup window make all entries concerning a certain channel. Name the channel (if you dislike Channel x), browse for your database (DBC) and select one (or more for a multiple Channel ECU) database nodes. For Gateways you have to add additional Channels.





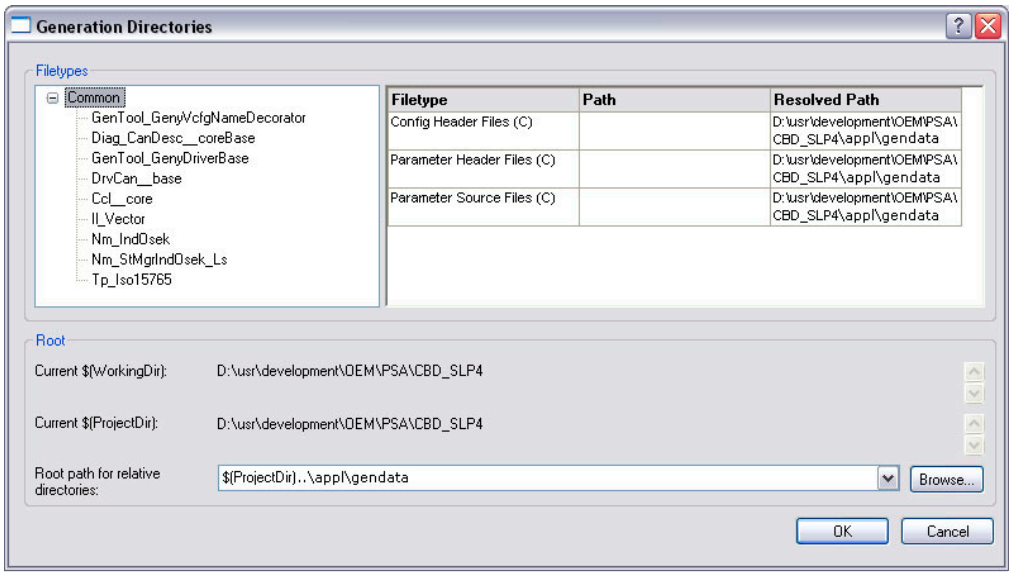
Use the Software **Component Selection** view to switch the necessary components on.

Now we could go on and adjust the path where GENy generates the files in.

Open the Generation Directories window via **Configuration / Generation Paths…**or use

Now save the configuration. This file will look like: *.gny.

Define paths for the generated files



➔ Enter the **Root path for relative directories**. Confirm with tab.

➔ To get the files for the different components generated in specific folders, add a relative path on the appropriate **Path** field for the component and for the **Filetype**.

You also could enter an absolute path in the **Path** field.

→ Always check your settings for all components under **Resolved Path** before confirming via **[OK]**.

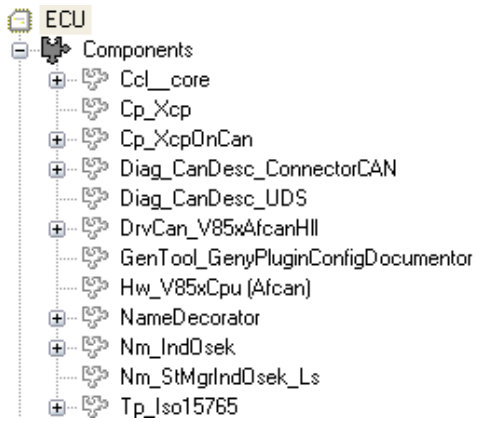
Info: The (C) behind **Filetypes** indicates that these are files in C language.

Now save the configuration (.gny file) again.

3.3.3 Settings for the CANbedded Software Components

Navigation View in GENy

The settings for the components are done via the Navigation and the Configuration view. Any component you have selected in the Software Component Selection view is listed in the navigation tree below Components



Let us go through all selected components and do the necessary settings.

Info: Remember that the channel is the major criteria where to find a configuration element of the component. E.g. the setting of the baud rate is in the component DrvCan_xxx and it is a channel specific setting.

The different components are

HW_XXXCPU

Some settings like the used derivative and compiler are very hardware dependent. They are selected in the HW_xxxCPU tree. Make sure that the derivative, the compiler and the register addresses are set correctly.

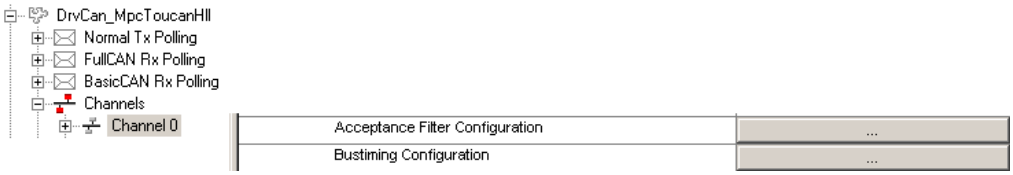
DrvCan_xxx

Here you configure the CAN Driver. Use the default settings for the first start up.

DrvCan_xxx / Channel / Channel X

The two most important settings here are the baud rate and the acceptance filters. Click on the buttons […] to open the corresponding windows and do the necessary settings (refer to GENy Online Help how to work with these two windows).

Filters and Bus timing



All further channel specific settings for the CPU are also hardware dependent.

NameDecorator

With the NameDecorator you can adjust the names of macros, functions and flags that are generated. Use the default settings for the first start up

Tp_Iso15765

Here you configure the ISO Transport Protocol. Use the default settings for the first start up.

Il_Vector

Here you configure the Vector Interaction Layer. Use the default settings for the first start up.

Inirect NM

Here you configure the Network Management. Use the default settings for the first start up

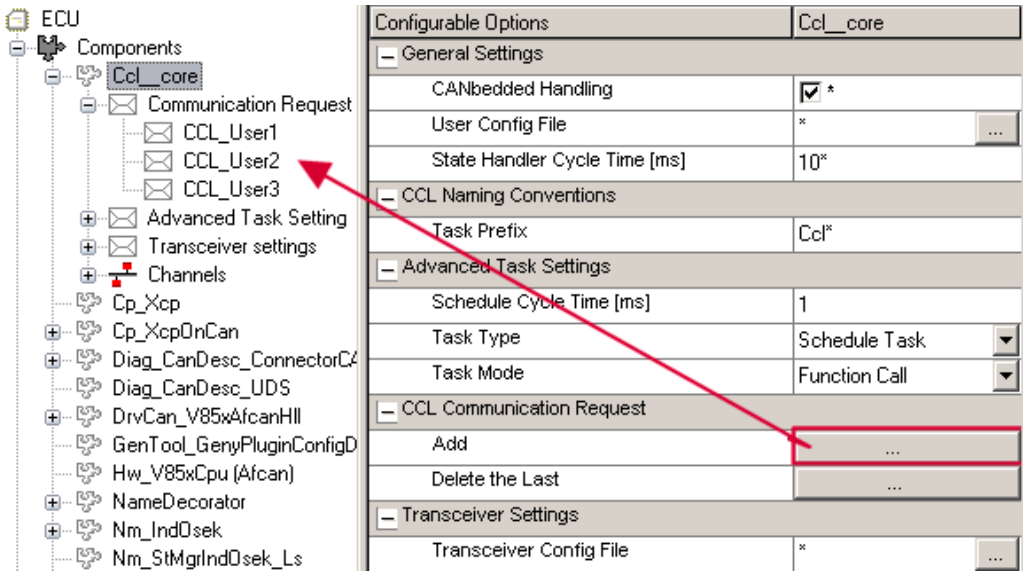
Diag_CanDesc_UDS

Here you configure the diagnostic component CANdesc. Use the default settings for the first start up.

Ccl_core – Communcation Control Layer

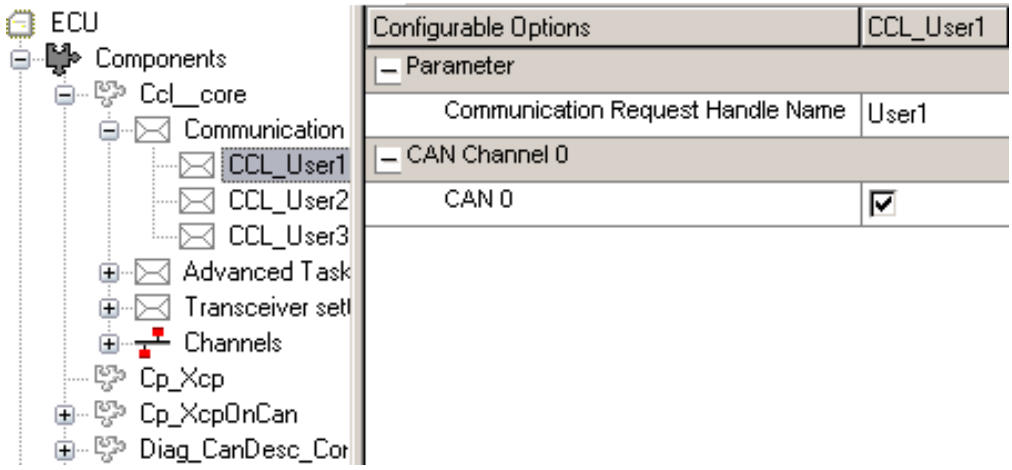
First open the Configuration View for the Ccl__Core. There will be no Communication Request available. You have to add this.

Click on the Add **[…]** button of the **CCL Communication Request** and a new entry in the Navigation view will be shown.



Name the Communictaion Request

One Communication Request appears without name, so you can only see the letter icon. Click on this empty icon and get the configuration view like in the following figure.

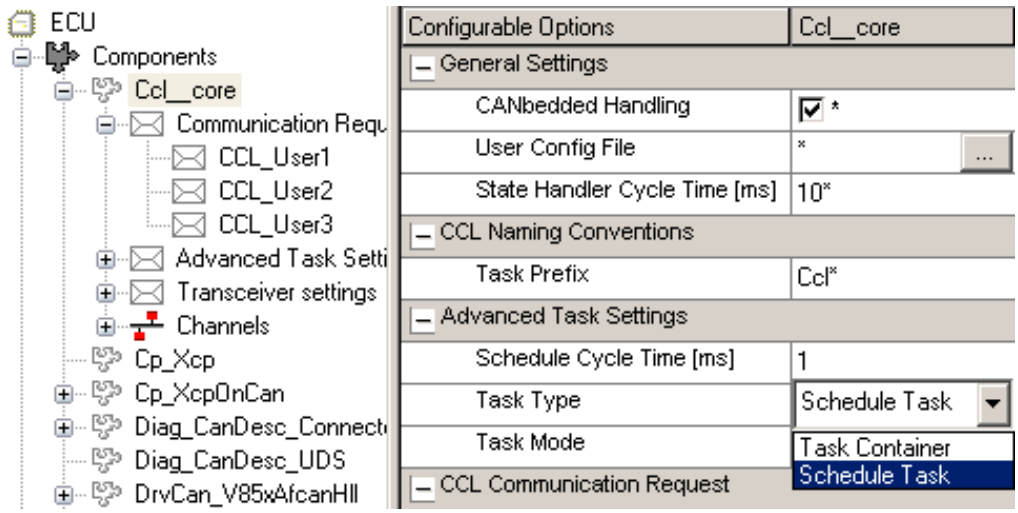


Enter the name for this Communication Request. In this case MyCommunicationRequest (CCL_MyCommunicationRequest: the configuration tool adds the CCL as prefix to the name.) is now the parameter for the functions to control the ECU states. We will use this parameter in the following software example, too.

Info: Make sure that the names for the requests are unique in the whole system. The names have to be ANSI C compliant.

Using CCL many things become easier for the application. So the complete initialization and the cyclic calls could be done by these components. Just check the box **[CANbedded Handling]** in the Ccl__core configuration view (see Figure 6 8).

Now we could switch to the CCL configuration view by choosing Ccl_core in the navigation view.



We switched the **CANbedded Handling** on, so we have to choose between the **Schedule Task** and the **Task Container** from the pull down menu. What does this mean?

We let the CCL handle the initialization and control of the components. Control means in this case the cyclic calling of all necessary CANbedded task functions and this could be done in two ways.

The easy way via Schedule Task

Schedule Task

The option **Schedule Task** is the easiest way for the application programmer. The function `CclScheduleTask()` will be generated and handles all cyclic CANbedded calls internally.

Your application has to call this single function and all component tasks will be called in the correct cycle time. The cycle time for this schedule task must be a common divisor of all selected CANbedded call cycle values.

See

Cross reference: [1] (see section Reference Documents on page 2).

for more details on Schedule Task concept.

Container

Task Container

The Task Container provides one function for all tasks with the same call cycle. The application has to call these tasks with the correct call cycle and need not to know which components are affected.

E.g. for all 10ms Tasks the task container function `Ccl_10_0msTaskContainer` is generated. See

Cross reference: [1] (see section Reference Documents on page 2)

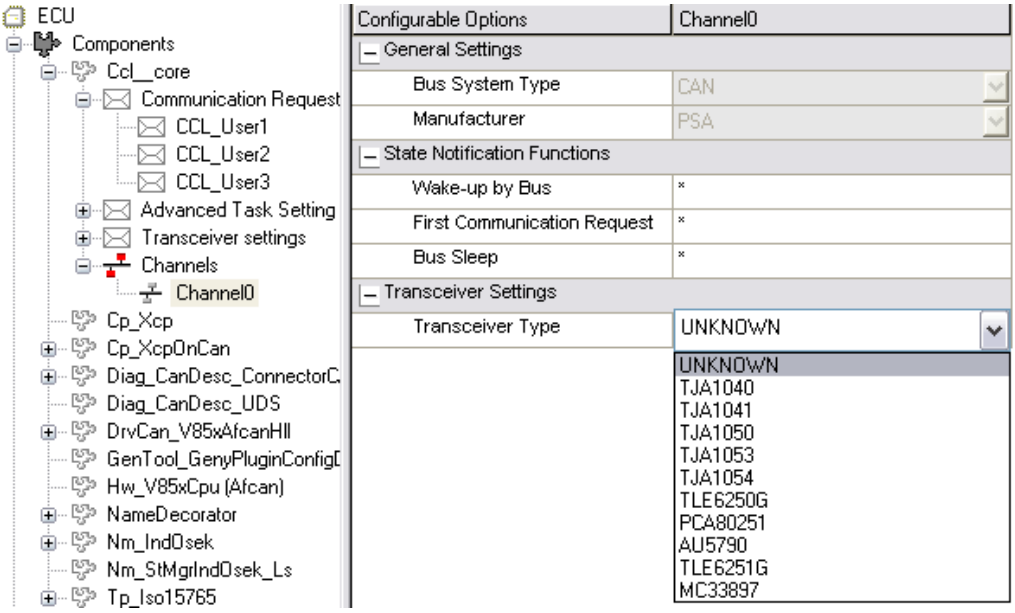
for more details on the Task Container concept.

Transceiver Settings

What is left now is to provide the CCL with the information concerning the transceiver you use. Open the channel specific configuration view of the CCL component as you see in the following figure. Use the pull down menu to select your Transceiver Type.

Info: GENy supports the most of the common transceivers. If your transceiver is not in this list, choose < Unknown >. For this case the application has to provide the transceiver handling via three callback functions.

Select your Transceiver



Transceiver Configuration File

After you have selected your Transceiver, you have to add your **Transceiver Config File**. This file includes information for additional transceiver coding.

Use the **[…]** button to load your **Transceiver Config File**.

Ports

Dependent on the transceiver you select there will be displayed a corresponding amount of ports, in this case the Enable Port, the Standby Port, the Error Port and the Wake Up Port.

Info: To do the following settings you need a deep insight in the hardware of your project.

Now you have to tell the CCL component how to control the ports of the transceiver. Please refer to

Cross reference: [1] (see section Reference Documents on page 2)

for more details how to set up the transceiver handling.

Transceiver Unknown

In this case you cannot do any further transceiver setting in the GENy as you see in the illustration below.

Unknown

For an unknown transceiver the CCL does not know how to handle the transceiver.

Transceiver

This has to be done by the application. For this reason the application is provided with 3 callback functions to handle this unknown transceiver:

`ApplCclInitTrcv` for transceiver initialization

`ApplCclWakeUpTrcv` for transceiver wake-up handling and

`ApplCclSleepTrcv` for transceiver sleep handling.

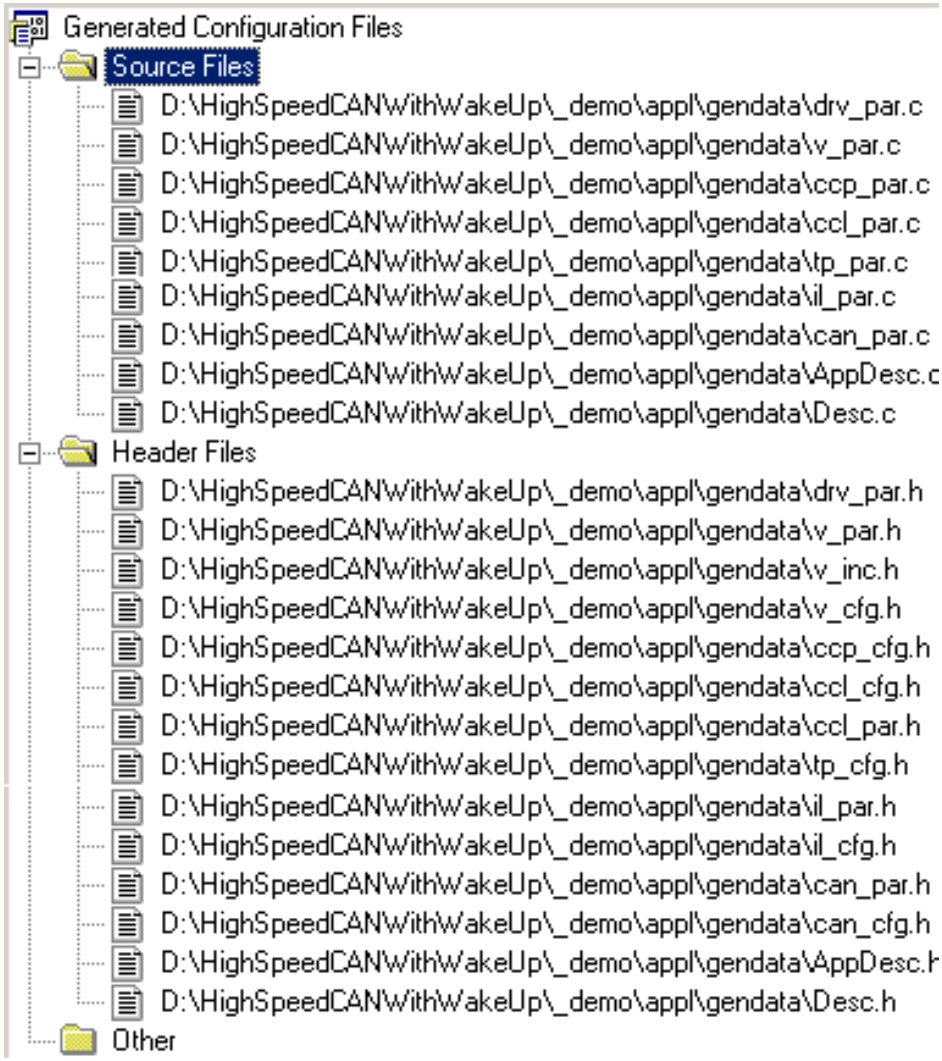
The code for these callback functions is very hardware dependent. Refer to the hardware manual of you transceiver for more information

3.4 STEP Generate Files

Let GENy generate the files to the appropriate, previously defined folders.

Now all configuration settings for the CANbedded software components are done and we could go on with the generation process. Click the icon Generate System.

Make sure the shown target paths are correct



In the **Output View / Generations** you see all generated files with the corresponding paths. The Generated Files from the System Configuration view show the generated

files in a sorted manner.

Info: Use this information to check if the files are generated in the desired paths.

The files for the CANdesc component are also generated, but via a different generator. So they are not displayed in one of these windows.

There are also generated XML files. These files are for internal use of GENy only and you must not change them.

Please check the output window also for (orange) warning and (red) error messages. If there are some, clarify the reason. For the final software setup, no warnings or errors shall be reported!

3.5 STEP Add CANbedded to Your Project

Add the CANbedded C and H files to your project or makefile.

Your Build environment

What to do in this step depends on your development environment. Perhaps you work with a makefile?

You have to add the files of CANbedded to your project. These are the files of chapter STEP Configuration Tool and DBC File on page 13 and the generated ones to the previous step.

Info: Always make sure that the path you generate the files in and the path your compiler is working on are the same!

At this point in time you are not able to compile and link the project. The files should be complete but there are several adaptations for you to do in your application.

Go on with the next step.

3.6 STEP Adapt your Application Files

Now your application files must be modified to use the CANbedded software components (includes, cyclic calls, initialization, callback functions).

Include, initialize and call the components cyclically.

Then connect CANdesc with your application.

Now all files for CANbedded and CANdesc are included in your project and we can go on to do the necessary adaptations in your application files.

These adaptations can be split in two categories:

- ➔ Include, initialize and do the cyclic calls for the CANbedded software components (use the component specific documentation for details).
- ➔ Connect the CANbedded software components to your application

As you use CCL this is very easy to do.

3.6.1 Including, Initialization and Cyclic Calls

Pre-compile

Including

If the Checkbox **CANbedded Handling** is checked then you only have to include two headers, the v_inc.h (that is generated) and the ccl_inc.h in this order. This header includes all necessary header files for the selected Software Components.

Info: If you do not use CCL CANbedded Handling you have to do the “includes” for all Software Components on your own.

Pre-compile

Initialization

With the CCL CANbedded Handling the Initialization becomes very simple.

`CclInitPowerOn();` is the only function for the initialization of all selected CANbedded Software Components.

Info: Without CCL CANbedded Handling all component tasks must be called in the correct order.

Call cyclic tasks

Cyclic Calls

The handling of the cyclic component task calls depends on the settings on the CCL configuration view (Schedule Task or Task Container see Figure 6 8). As mentioned above you only have to call one function if you select Schedule Task:

`CclScheduleTask();` make sure that the call cycle is a common divisor of all tasks call cycles

In case of the Task Container you have to call all existing task container functions. These functions look like this:

```
Ccl_10_0msTaskContainer();
```

Make sure that you call all these tasks, each with the correct call cycle and offset.

Info: The functions to control the cyclic task calls are generated in the file ccl_par.c. Especially when using the task container functionality, refer to this file to get all generated functions.

Info: If CCL CANbedded Handling is not checked, your application must do all the task calls for the CANbedded software components on its own and in the correct cycle time.

3.6.2 Application Handling of User Requests and the Bus Communication

As mentioned in the chapter Requesting and Releasing Bus Communication on page 36 the users can request or release the bus communication. Remember to control the modes via your application and the functions:

```
CclRequestCommunication( CCL_MyCommunicationRequest )
CclReleaseCommunication( CCL_MyCommunicationRequest )
```

The parameter is the handle (you configured in the configuration tool) with CCL_ as

prefix.

3.6.3 CANbedded Software Component Callback Functions

Callbacks are still missing

At this point in time you would be able to compile but not to link. The callback functions of the components CANdesc and CCL are still missing. The detailed description of these callback functions is given in the TechnicalReference of each component. In the following we only show the callback functions and introduce the templates to be able to compile, link and run the system basically.

Diagnostic callbacks

Callback Functions for Diagnostics

Any diagnostics service needs at least one (maximum three) callback function (Handlers, see the documentation for CANdesc). Dependent on the amount of diagnostic services and their settings there are different callback functions. You find the generated prototypes for the diagnostic callback functions in `appdesc.h`.

The diagnostic callback functions are provided in the generated template file called `ApplDesc.c`. Add this template to your project.

Info: Remember to fill the callback functions later. This is only to be able to get a basically working system that is the base for further development.

Read the comments in the template carefully. This will help you better understanding how to use the template.

CCL callbacks

Callback Functions For Communication Control Layer (CCL)

There are several callback functions to get the CCL running. The detailed description of how to handle these functions correctly is in the

Cross reference: TechRef CCL

This is the list of all necessary callback functions:

```
vuint8 ApplCclCanStandby( vuint8 sleepResult )
{
  return kCclNoRepeatCanSleep; /*is defined to 0*/
}
```

Info: This callback function is already part of the CCL template.

```
void ApplCclTrcvGoToSleepWaitTime( void ){}
```

Info: This function is only necessary for some transceiver derivatives

```
void ApplCclWakeUpTrcv( void ){}
```



```
void ApplCclSleepTrcv( void ){}
void ApplCclInitTrcv( void ){}
```

Info: These functions are only necessary in case of you have selected UNKNOWN as Tranceiver Type on the CCL configuration view.

```
void ApplCclBusOffStart ( void ){}
void ApplCclBusOffEnd( void ){}
```

Info: The application will be informed in case of a bus off.

```
void ApplCclInit( void ){}
```

Info: Remember to fill the callback functions later. This is only to be able to get a basically working system that is the base for further development.

3.7 STEP Compile and Link your Project

Compile and link the complete project and download it to your test hardware or development environment.

It is almost done!

Now we have all includes, all initializations, the components do have the cyclic calls of their task functions and all callback functions are provided (but not programmed yet).

Start the compiler or makefile and get the project compiled and linked.

Is it ok? No errors?

Congratulations, that's it!

Go on to the next step and do the testing.

3.8 STEP Test it via CANoe

Via an appropriate Tester (e.g. CANoe) check the results.

WELL DONE!

The last step is to test what we have done until now. This is no detailed test. It is just to make sure that we have bus communication.

The ideal tester would be CANoe from Vector Informatik. Connect your hardware to your CANcard, start CANoe, set the correct baud rate and have a look at the trace window.

You see any bus communication? No error frames?

Congratulations!

The basic step is done, the CANbedded software components are basically working together with your application.

That is the base you can start from to optimize your system. Remember the callback functions we left empty and refer to

Cross reference: [1] (see section Reference Documents on page 2)

for more detailed information.

3.9 STEP Test and Release Hints

Special Caution: Before the SOP it is your duty to recalculate the automatically generated values for the baud rate and the acceptance filters. Make sure you have the current version of the necessary hardware description available to validate the used (CAN) controller settings.

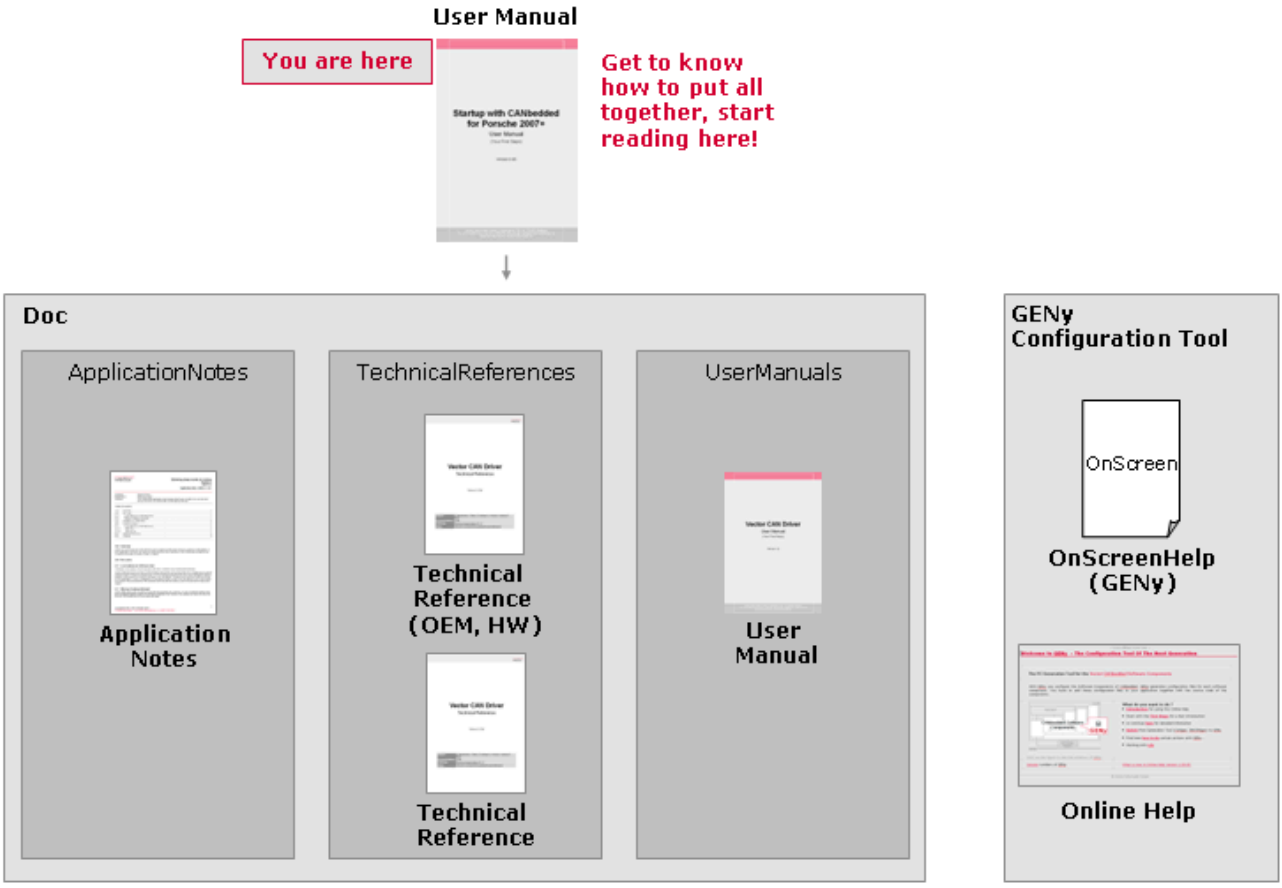
4 Basic Information

In this chapter you find the following information:

4.1 Documentation Structure for CANbedded Components

Your delivery

To be able to get along with the delivery fast and easy this chapter will give you an overview of the delivery and its documents.



Always read the User Manual first

The first document you should read is this one, the startup user manual. It gives you an overview of all components and of how to put this stuff together and get it to work basically.

Components

The delivery contains different components like Interaction Layer, CAN Driver, etc. A component is normally documented in the following way.

Caution: If you use the CCL component, read its manuals and references first after this user manual. CCL makes the handling of the other components very simple.

Different kinds of documents

UserManual_<ComponentName>

Very easy entry in the software component. It contains a step-by-step introduction of how to use this component. This is not available for all components.

TechnicalReference_<ComponentName>

Very detailed description of the component and its functionality. It gives you a very

technical sight on the component.

TechnicalReference_<ComponentName>_OEM/HW

Some features of the component are very hardware-dependent or differ from one OEM to the other? Then this kind of document is there to explain these special technical aspects.

Application Note

Very special topics of the component, especially in connection with other components are described in application notes. Not all components do have application notes.

GENy – Configure your components

The configuration tool GENy provides two major ways of documentation. The OnScreen Help view and the Online Help.

Online Help

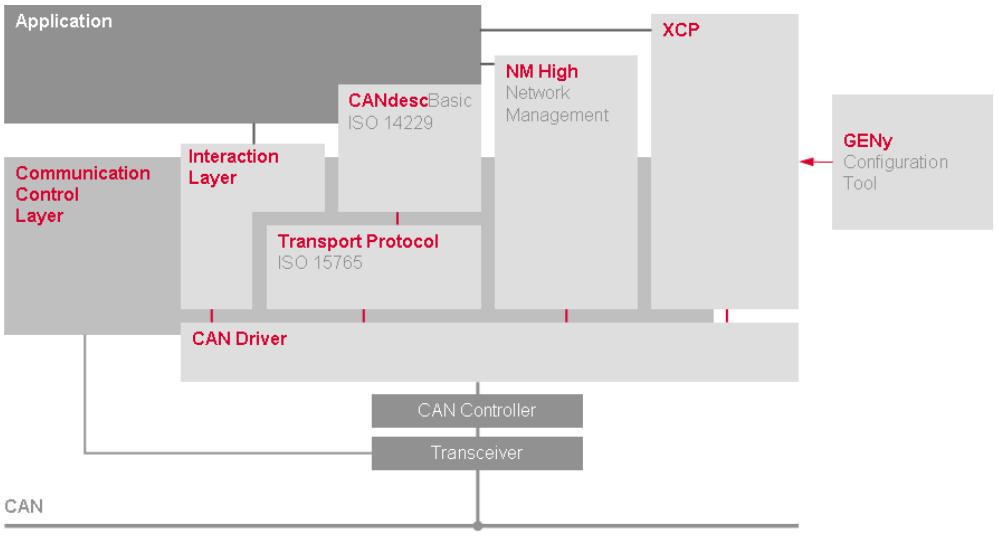
The GENy Online Help gives you information about the generic handling of GENy. It explains the different GENy windows, shows how to work with the GUI and offers How-To-Descriptions for the frequently asked questions like how to set up a configuration, how to update the database file, etc.

OnScreen Help

This is an area in the GENy GUI where specific information about the presently activated configuration element is displayed. This provides you with the information you need to have to decide whether to use a GENy option or not. The more detailed description and technical background can be found in the technical references of the configured component (see above).

To guarantee quality it is obvious to use standard software components for the communication part of your ECU. The software components of Vector Informatik are called CANbedded and are shown in the figure below via grey boxes with red inscription. The application is on top level and uses the functionality of the CANbedded software components for Communication and Diagnostics.

CANbedded Layers



Hardware Interface

CAN Driver

The CAN Driver handles the hardware specific CAN chip characters and provides a standardized application interface. The CAN Driver for the PSA can transmit and receive standard (11 bit) CAN Ids.

Transmission modes and more…

Interaction Layer

The Interaction Layer (IL for short) is responsible for the transmission of messages according to specified rules, monitoring receive messages, timeout monitoring, etc. It provides a signal oriented application interface for the application.

Break the limitation of 8 data bytes

Transport Protocol

The CAN protocol is restricted to 8 data bytes per message. But in some cases (e.g. diagnostics) you need to exchange much more than 8 data bytes. The segmentation of the data, the monitoring of the messages and the timeout handling is done by the Transport Protocol (TP for short).

UDS

Diagnostics

The diagnostics layer (DIAG for short) works according to ISO14229 (UDS) and PSA specifications.

Control transition to bus sleep state

Network Management

The Network Management (NM for short) is the component to control the bus, to synchronize the transition to bus sleep, error recovery after bus-off, etc.

Measurement and Calibration

Universal Measurement and Calibration Protocol (XCP)

This is the Software Component for measurement and calibration on several bus systems. To mention some feature: read and write access to various memory locations or flash programming. XCP is not part of the official CANbedded delivery process to PSA. See chapter Availability and Usage of XCP within the CANbedded Stack on page 37 within the CANbedded stack for more details.

The powerful component

Communication Control Layer (CCL)

This component provide the following features: communication bus access, encapsulation of network management handling, handling of the CANbedded stack, control of the transceiver and the handling of the start-up delay time.

4.1.1 Configuration Tools and Files

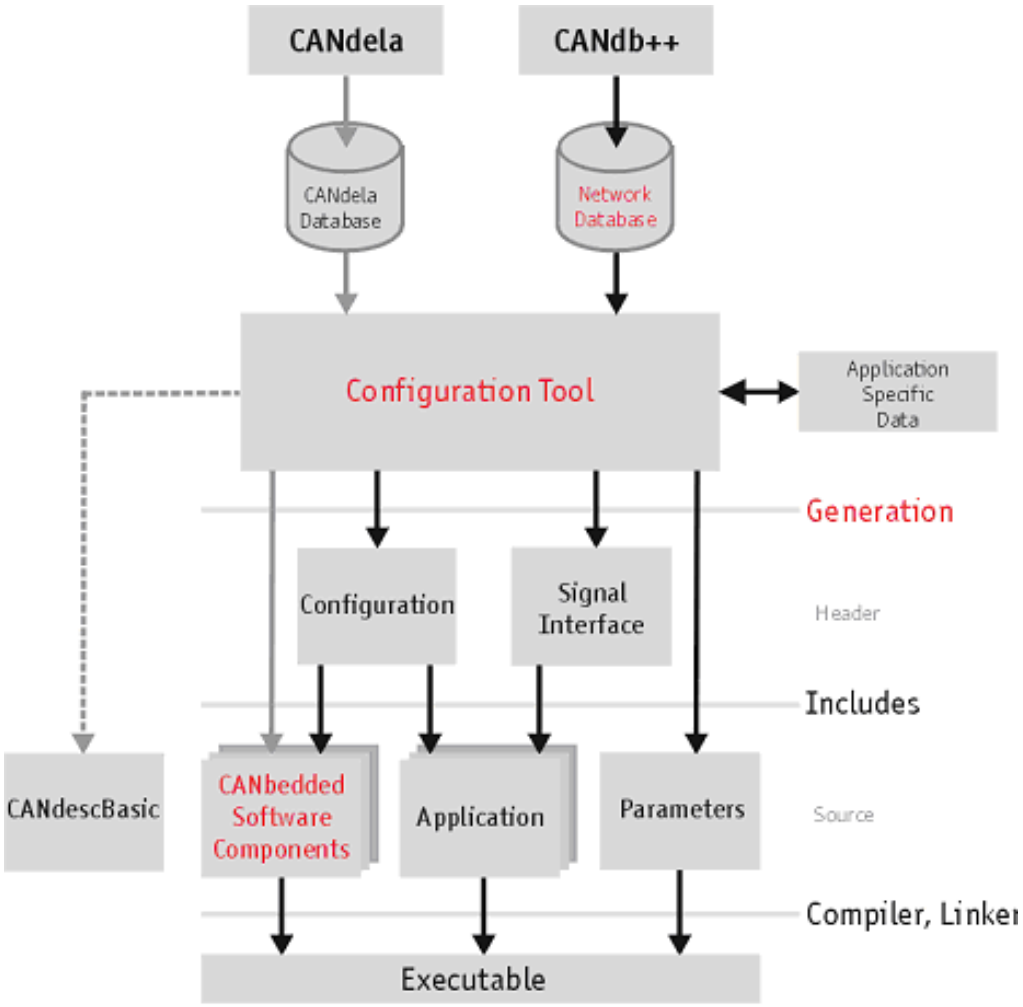
Configuration tools and files

The delivery contains the MS Windows based configuration tool GENy. It is used to adapt the delivered source code via project specific generated data and configuration defines to your project needs.

There is always the same concept behind the usage of Vector´s CANbedded software components (red labled boxes).

The standard generation process for Vectors Software Components.

CANdesc is a completely generated Software Component.



GENy combines the information of the network database (DBC file) and your application specific configuration for the CANbedded components. As a result it generates files (header: configuration files and signal interface, source: parameters) containing the upper information. You have to compile/link these files together with the source code of the CANbedded software components and your application files.

Derived from this concept there are three sorts of files:

- Generated files (Configuration, Signal Interface, Parameters) you must not change manually because the next generation process will delete your changes (e.g. can_cfg.h). Changes can only be done via the configuration tool.
- Files that form a component (CANbedded software components) and must not be changed at all (e.g. can_drv.c).
- Files for your application.

Info: The network database (**DBC file**) is provided by Vector Informatik if it is ordered or must be created as active work of TIER1. Each CAN bus has its own DBC file describing the communication for that system.

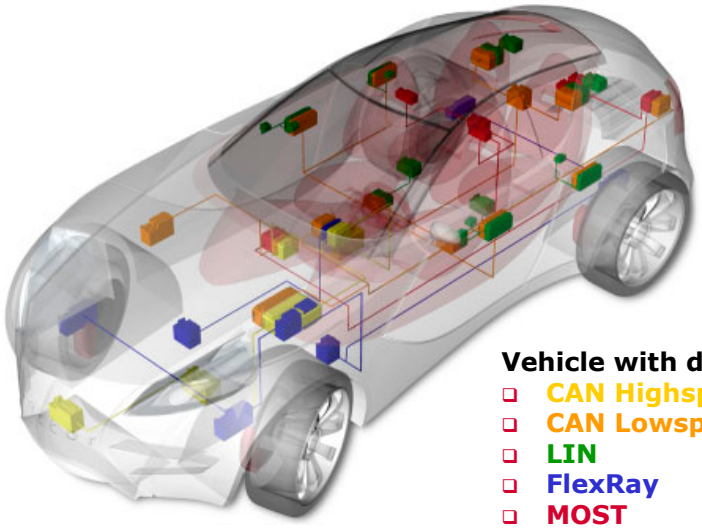
The DBC file can be viewed and also edited by the tool CANdb++. CANdb++ is part of each CANoe/CANstress/CANscope package but can also be purchased separately as a CANdb++/Admin version.

The diagnostic description for CANdescBasic is part of the CANbedded delivery

4.2 An Overall View

ECU in the focus

What we are now talking about is an ECU, a module to be built-in a vehicle like shown in the figure below. Almost every ECU participates in a certain bus system like e.g. CAN, FlexRay or LIN.

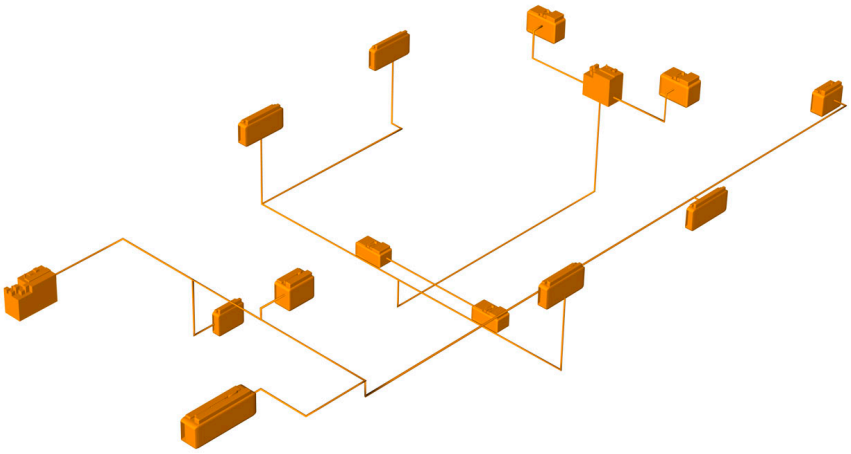


Vehicle with different bus systems

- CAN Highspeed
- CAN Lowspeed
- LIN
- FlexRay
- MOST

So any ECU within one bus system has to provide an identical interface to this bus system because all ECUs have to share information via this bus system as you see in the figure below.

CAN Lowspeed as an example bus system



For that reason all ECUs are built-up in the same way. There is a software part to realize the main job (application) of this ECU e.g. to control the engine or a door. The other part is the software part to be able to communicate with the other ECUs via the bus system that is the communication software.

4.3 An ECU – a More Detailed View

4.3.1 Generic Usage of CANbedded Software Components

Initialization, cyclic calls and callback functions.

To get almost all components to work your application has to initialize the components and call certain component task functions cyclically. This is to derive the time base each component needs. You can adjust the cycle times on the component's configuration views of configuration tool GENy.

No rule without exception:

For some components there is more than one task to be called within a predefined cycle (e.g. Interaction Layer: Rx and Tx task).

The CAN Driver has no task to be called (unless used in polling mode) because it is an event driven component and uses interrupts.

While the components are running there are different callback functions for the application to control the components' behavior.

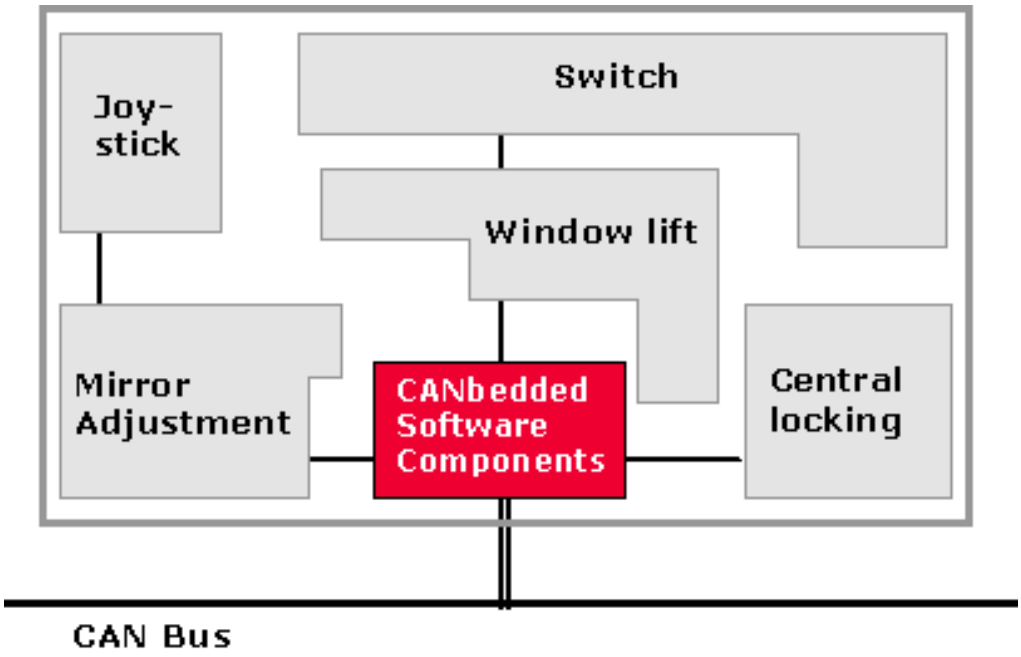
The PSA CANbedded stack contains usually the CCL (Communication Control Layer), which will ease up the integration task. Please refer to

Cross reference: [1] (see section Reference Documents on page 2)

for more details.

4.3.2 Independent Software Components in an ECU

N:1 relationship



A typical ECU application is divided up into multiple functional blocks. Each functional block has its own tasks and might be implemented by another programmer. To ease up independent and decoupled development of application code, the CANbedded stack provides independent network control and signal access API's for each functional block.

4.3.3 Requesting and Releasing Bus Communication

Bus communication (network control) is requested and released using CCL APIs. If a functional block needs bus communication, a communication request is set via CclRequestCommunication. If no bus communication is needed in the functional block, it releases the communication request via CclReleaseCommunication.

Please note that the request and release is asynchronous. I.e. there is a delay between requests and availability of network communication. The network has to be handled after a release request until it is really stopped (state BusSleep).

4.3.4 Multiple Channel ECU

More than one physical channel

Multiple channels ECU means to control more than one physical CAN channel. For each channel you need a database. To generate for a multiple channel ECU, add as much database as channel to your configuration tool setup.

Make sure that you do the necessary settings in the tool for each channel. Below any component there is a channel tree that contains the settings for the different channels.

4.3.5 Availability and Usage of XCP within the CANbedded Stack

Part of the CANbedded delivery for PSA

XCP is part of the official CANbedded delivery process to PSA.

4.3.6 Start-up Time of the CANbedded Stack

The CANbedded software components usually need very little time for initialization due to mainly data structures in the RAM are initialized.

The hardware specific drivers like the CAN or LIN driver access also hardware registers of the microcontroller. The runtime for this depends strongly on the number of registers, the speed of the internal µC data bus and, depending on the µC, the necessary algorithm to access and control these registers (e.g. requesting the access and waiting till the internal µC bus grants the access).

All in all the runtime of the CANbedded stack initialization highly dependents on the configuration (e.g. how many messages have to be initialized, what does the application do within the used callback functions, etc.).

Info: Please refer to the resource documentation Excel sheet stored in the _doc directory of the delivery for details about your specific delivery package.

Power Supply

PLL

EEPROM

RAM

SPI

Some typical, time-consuming actions during start-up outside of the CANbedded stack are :

- ➔ Switching on the power supply and wait till it is stable: a few [ms].
- ➔ Switching on the PLL and wait till it is stable: [µs] up to few [ms].
- ➔ Access to serial EEPROM (internal and external): can be very time consuming
- ➔ RAM check: can be very time consuming

Initialization of external hardware via SPI: can be very time consuming

4.3.7 Resources of the CANbedded Stack

The CANbedded stack is optimized for minimum RAM and ROM usage and can therefore be widely configured to the very special needs of your project.

Any supported but in the concrete project unused feature can be disabled via compiler switches during compile time so that this functionality does not consume RAM, ROM and runtime. It is therefore difficult to give an exact amount of resource needs for the concrete project upfront.

5 Further Offers

In this chapter you find the following information:

5.1 Hotline

Hotline

The hotline is available by email or by phone from 13:00 to 16:00 (German time).

Please **contact**: Vector Informatik GmbH, 70499 Stuttgart

email: Embedded@vector-informatik.de

phone: +49 711 80670-400

5.2 Training Classes

Training

Vector provides training classes for CAN, CAN tools, CANbedded, PSA specific topics and also other bus systems like LIN and FlexRay. Please contact your technical contact person at Vector for more information to this topic.

5.3 Integration Support

Support

Vector provides on-site integration support for the PSA CANbedded stack into your application. Please contact your technical contact person at Vector for more information on this topic. The integration support is an optional service for PSA projects and shall be performed during project start.

5.4 Integration Review

Reviews

Vector provides on-site integration review for the PSA CANbedded stack into your application. Please contact your technical contact person at Vector for more information on this topic. The integration review is according to PSA documents a recommended service for PSA projects and should be performed before the project is finalized.

6 Additional Information

In this chapter you find the following information:

6.1 Persistors

What is the Persistor for?

The CANdela data base file (CDD) is created by CANdela Studio and used by GENy for configuring CANdesc or DCM/DEM in AUTOSAR environment.

If you use a newer version of the CANdela Studio, the format of the CDD file could be also newer than your GENy is able to deal with.

The Persistors are responsible to convert the newer CDD file into a CDD file which is able to read by GENy.

Update Persistors – Download current Version

The latest Persistors can be downloaded from Vector homepage

www.vector-worldwide.com.

Select **Downloads** and then the three settings for **Products**, **Categories** and **Standards**.

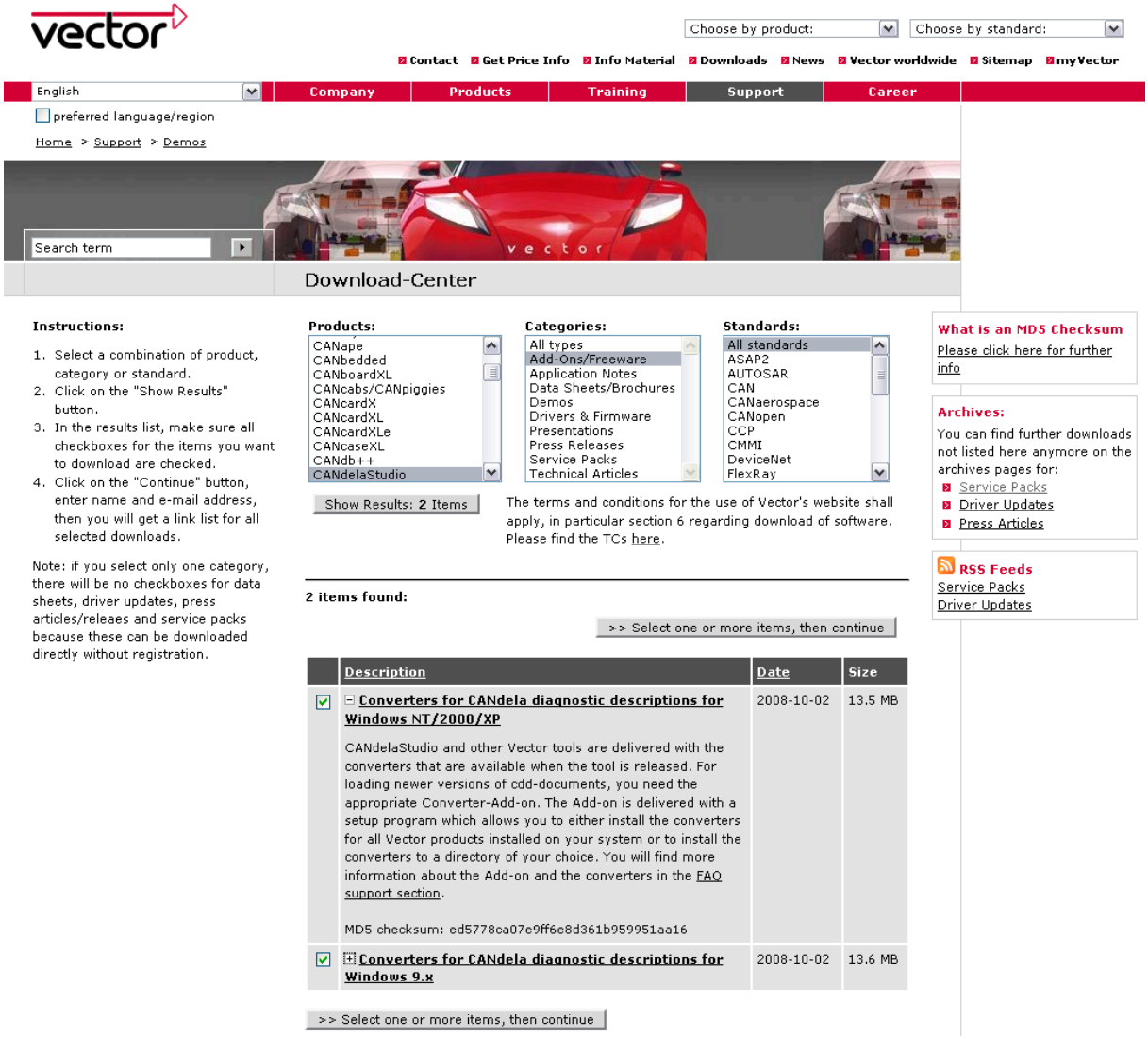
- → Products: CANdela Studio
- → Categories: Add-Ons/Freeware
- → Standards: All Standards

Cross reference: See the following illustration.

Available for NT/2000/XP or Windows 9.x

The name for the Persistors download is:

- → **Converters for CANdela diagnostic descriptions for xxx**.



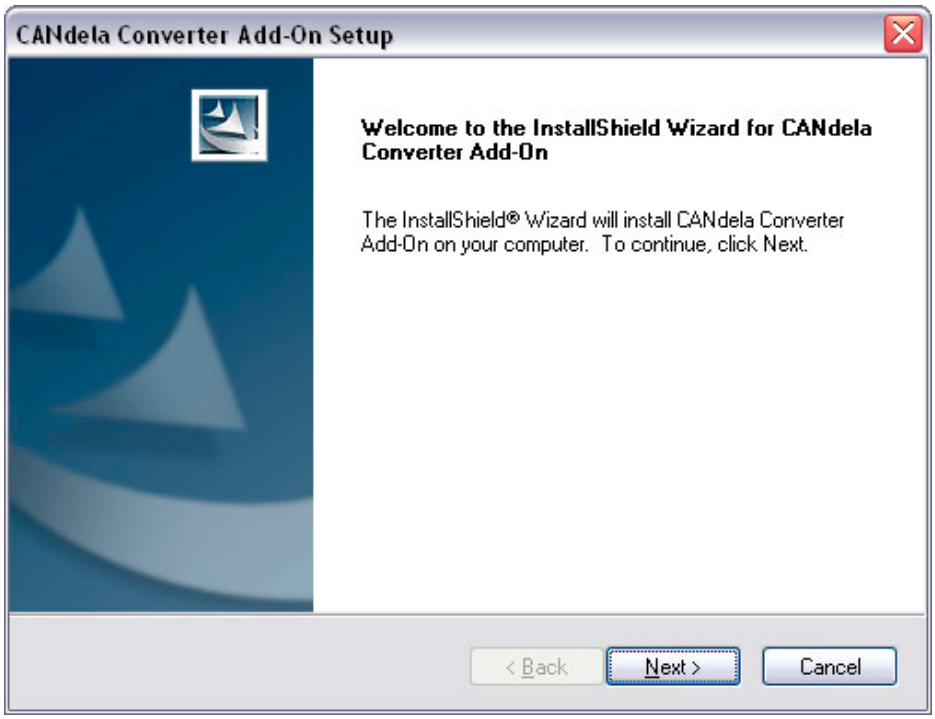
Download

Select on or more items from the list (☑) and click on **[>> Select one or more items, then continue]** to download the files after entering some administrative information.

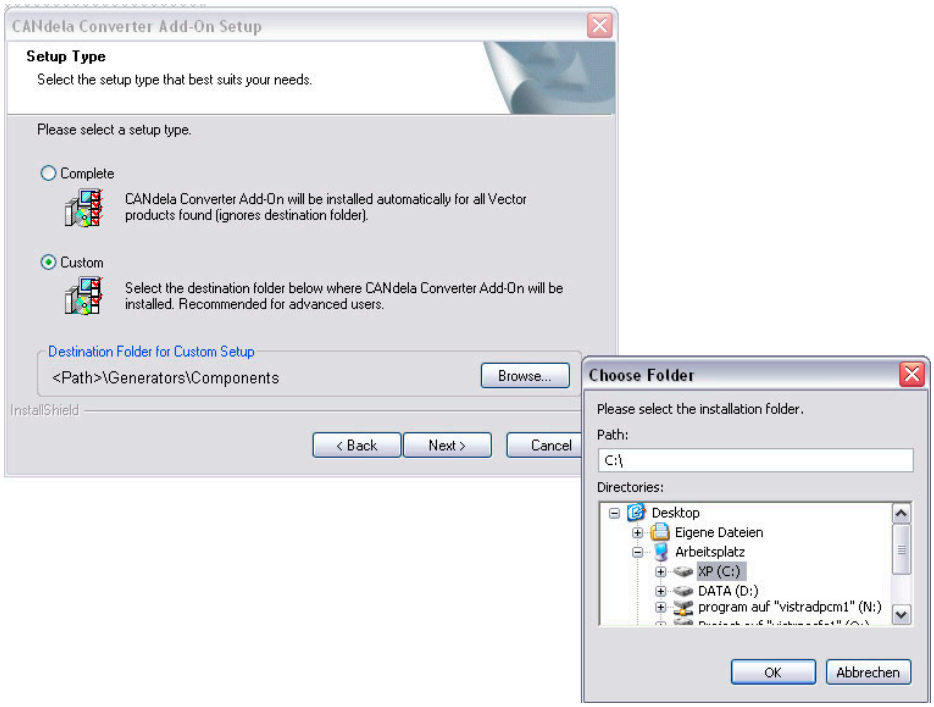
6.1.1 Update Persistors – Install current Version

Follow description step by step

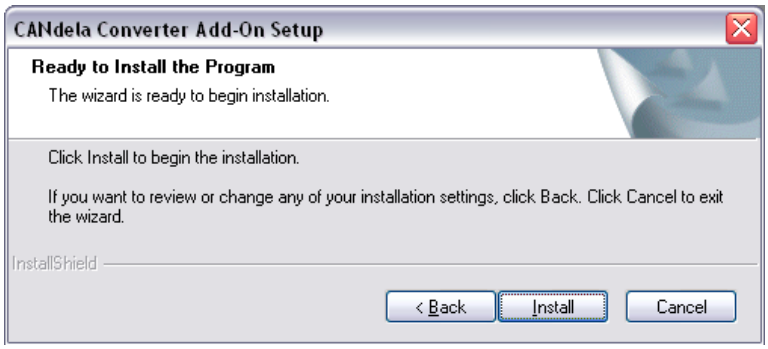
Start the downloaded file **SetupPersistorsXP.exe**.



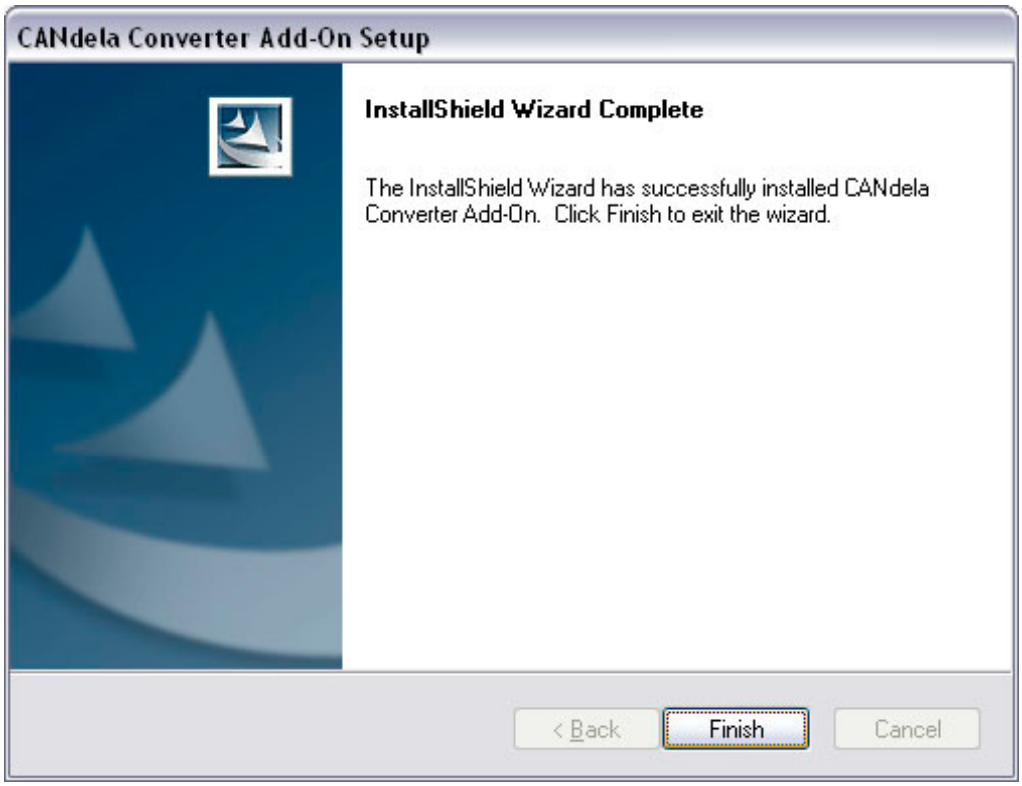
Click **[Next]**.



Select **Custom** and enter the path to the **…\Generators\Components** folder as **Destination Folder for Custom Setup** and click **[OK]**.



Click **[Install]** and the installation process will be started and then on **[Finish]** when ready.



Ready

Now the current Persistors are installed and your GENy is able to read the latest CDD file.

7 FAQs

In this chapter you find the following information:

7.1 Introduction

Find not search

You have a certain question? You just want to know how to do e.g. a certain setting without reading the whole document again?

Then go on reading the following list and use the links to get at the place in the document where your question will be answered.

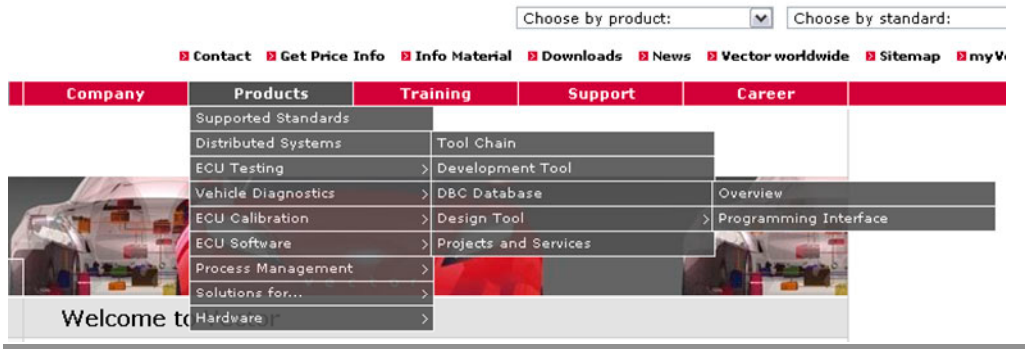
This chapter will be extended continuously.

7.2 Frequently Asked Questions

FAQ: You need a DBC file for your project, wherefrom do you get it?

You have two possibilities to get a DBC file for your project. You can order the DBC file from Vector (Vector create a new DBC file based on your specification) or you can create a DBC file with the CANdb++ Editor by yourself.

For more information about the CANdb++ Editor have a look at **www.vector.com**. (see Products / Distributed Systems / DBC Database / Overview)



FAQ: How to start GENy in a comfortable way with already loaded configuration?

There are different ways to start GENy and to load a configuration. The comfortable ways of using GENy is via a link or a batch file (see section Start GENy with a Link or a Batch File on page 13).

FAQ: The transceiver type is unknown, so what transceiver settings you have to set in GENy?

You only have to set transceiver type as unknown (see section **Transceiver Unknown** on page 22).

8 Address table

Vector Informatik GmbH

Vector Informatik GmbH
Ingersheimer Str. 24
D-70499 Stuttgart
Phone: +49 (711) 80670-0
Fax: +49 (711) 80670-111
mailto:info@de.vector.com
http://www.vector-informatik.com/

Vector CANtech, Inc.

Vector CANtech, Inc.
Suite 550
39500 Orchard Hill Place
USA-Novi, Mi 48375
Phone: +1 (248) 449 9290
Fax: +1 (248) 449 9704
mailto:info@us.vector.com
http://www.vector-cantech.com/

Vector France SAS

Vector France SAS
168, Boulevard Camélinat
F-92240 Malakoff
Phone: +33 (1) 4231 4000
Fax: +33 (1) 4231 4009
mailto:info@fr.vector.com
http://www.vector-france.com/

Vector GB Ltd.

Vector GB Ltd.
Rhodium Central Boulevard Blythe Valley Park
Solihull, Birmingham
West Midlands B90 8AS
Phone: +44 121 50681-50
mailto:info@uk.vector.com
http://www.vector-gb.co.uk

Vector Japan Co., Ltd.

Vector Japan Co., Ltd.
Seafort Square Center Bld. 18F
2-3-12, Higashi-shinagawa, Shinagawa-ku
J-140-0002 Tokyo
Phone: +81 3 (5769) 7800
Fax: +81 3 (5769) 6975
mailto:info@jp.vector.com
http://www.vector-japan.co.jp/

Vector Korea IT Inc.

Vector Korea IT Inc.
Daerung Post Tower III, 508
Guro-dong, Guro-gu, 182-4
Seoul, 152-790
Republic of Korea
Phone: +82(0)2 2028 0600
Fax: +82(0)2 2028 0604
mailto:info@kr.vector.com
http://www.vector-korea.com/

VecScan AB

VecScan AB
Theres Svenssons Gata 9
SE-417 55 Göteborg
Phone: +46 (31) 76476-00
Fax: +46 (31) 76476-19
mailto:info@se.vector.com
http://www.vecscan.com/

9 Glossary

Network Management.	The Network Management manages the availability of different networks on a channel. It is responsible for the synchronized transition between the communication states ‘sleep’, ‘prepare sleep’ and ‘active’ for all modules. Network Management serves to ensure the safety and availability of the communications network of autonomous control units. The OSEK NM distinguishes between node-related (local) activities, e.g. initialization of the node and network-related (global) activities, e.g. coordination of global NM operating modes.
Transport Protocol	Some information that must be transferred over the bus does not fit into individual message frames because the data length exceeds the maximum of 8 bytes. In this case, the sender must divide up the data into a number of messages. Additional information is necessary for the receiver to put the data together again.
Multiple Channel ECU	A multiple channel ECU is connected to several busses. So it is possible to transmit signals to and receive signals from all modules on these busses.
OSEK	Open systems and the corresponding interfaces for automotive electronics

10 Index

Y

Get more Information!

Visit our Website for:

> News

> Products

> Demo Software

> Support

> Training Classes

> Addresses

www.vector-worldwide.com