Welcome to CANbedded

Implement Vector's Delivery.
Within few steps from
<delivery>.exe to <application>.hex

V2.0   2006-09-13

These slide show you how to use a CANbedded software component delivery beginning with the installation until a first successful test run.
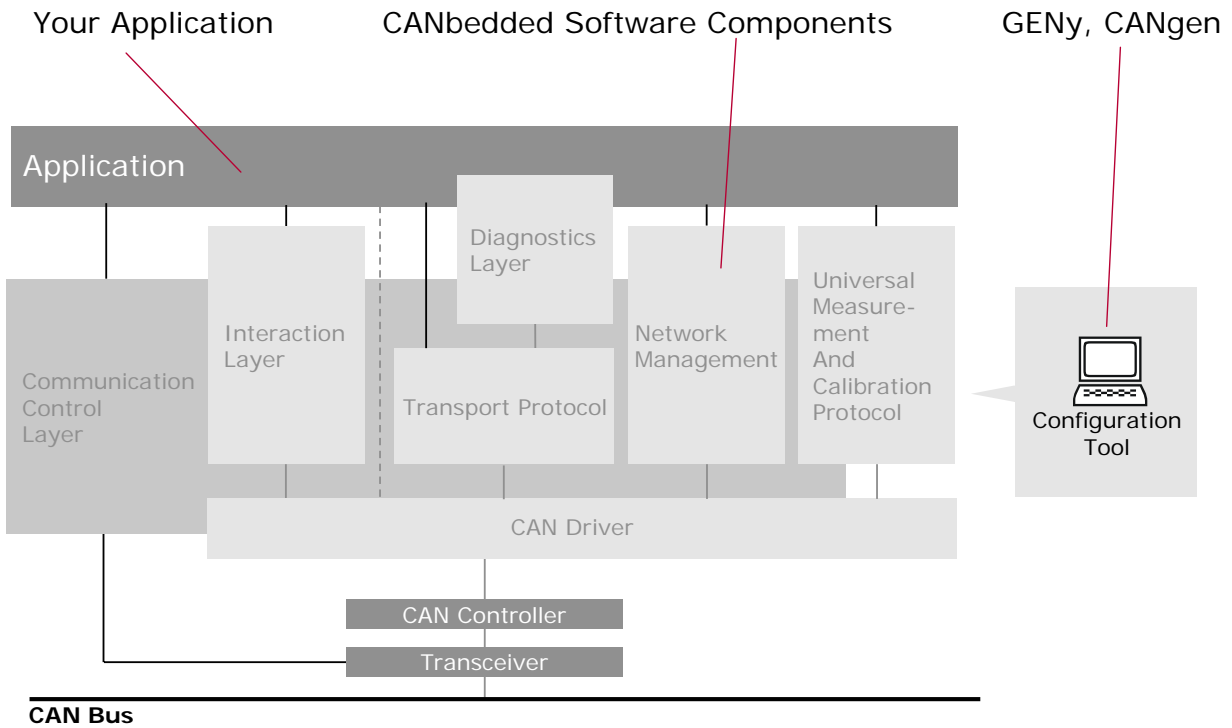
> Introduction

Components and Project Folders

Delivery

Step by Step

Summary

Your Application   CANbedded Software Components   GENy, CANgen

CAN Bus

The illustration shows the layer model of the CANbedded components, their basic functions and connections. **CANbedded** consists of a set of source code components (CANbedded Software Components) you have to include in your application. The sort of components depends on your **delivery**.

The **Configuration Tool** is the connection between the components and your project specific needs. It generates files you also have to include in your application.

Configuration Tool
for parameters and configuration of all components
more see Online Help

Communication Control Layer
Software component integration and hardware abstraction.

CAN Driver
Hardware specific CAN controller characteristics and provision of a standardized application interface

Interaction Layer
with signal interface

Network Management
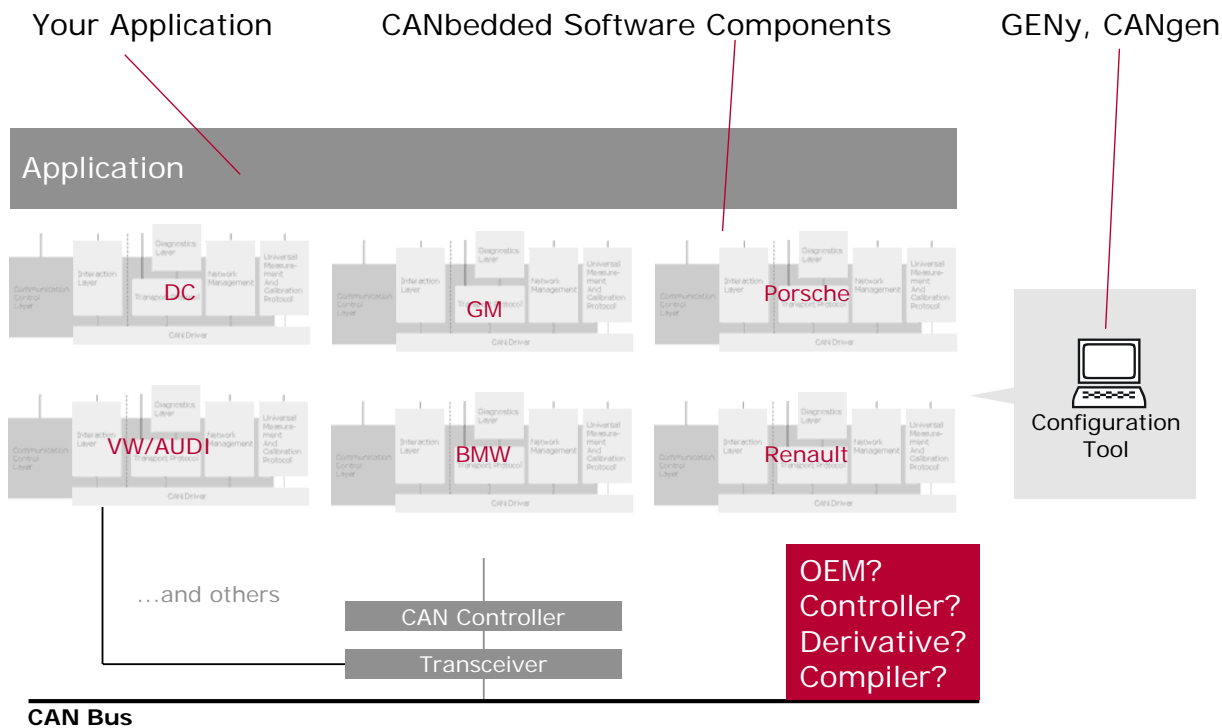to control the CAN ECU

Transport Protocol
for data exchange of more than 8 data bytes

Universal Measeurement and Calibration Protocol
Measurement and calibration of the ECU via different bus systems.

Diagnostics Layer
according to Keyword Protocol 2000 / UDS

Introduction

Your Application   CANbedded Software Components   GENy, CANgen

Application

DC   GM   Porsche

VW/AUDI   BMW   Renault

Configuration Tool

…and others

CAN Controller

Transceiver

OEM?
Controller?
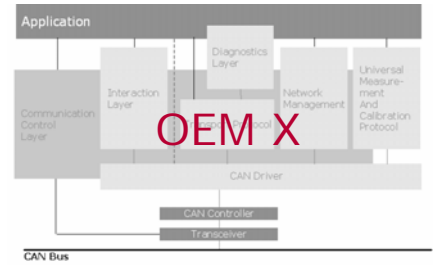Derivative?
Compiler?

CAN Bus

Dependent on the OEM the delivery and its content can differ.

Some components do a similar job but have a different name and a different API, e.g. the Interaction Layer is different for DaimlerChrysler, BMW and other OEMs.

This Delivery is:

Tailored for:

- ❑ Controller
- ❑ Derivative
- ❑ Compiler

Can be used for*:

- ❑ Projects for one vehicle manufacturer
- ❑ Can be used for multiple projects or car lines for this vehicle manufacturer

*right of usage can differ, for specific information refer to the quotation

The delivery is tailored according the questionnaire you filled-out. So the delivery contains all components you need tested for you controller/derivative/compiler.

You can use the delivery not only for one project but for all other projects for this vehicle manufacturer for different car lines.

Basic concept is almost the same for all OEM

❑ C code CANbedded Components

❑ Configuration Tool

OEM-specific differences in

❑ Component combinations

❑ Components

❑ API

vector

CANbedded component are source code components in C. You can tailor the huge amount of functionality of each component by using the configuration tool.
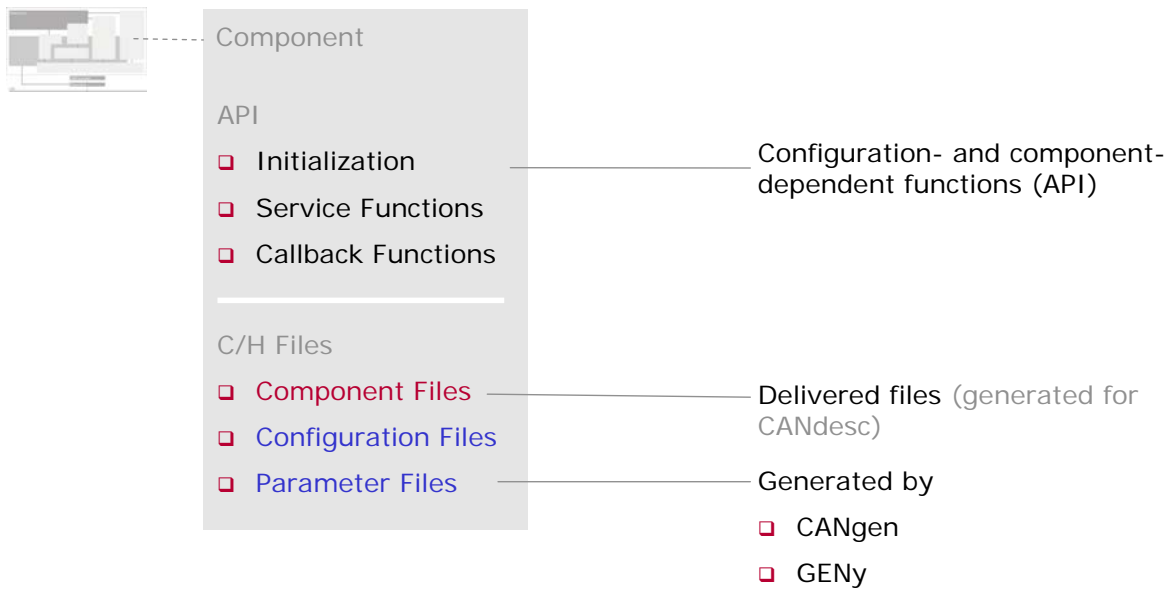
Introduction

> Components and Project Folders

Delivery

Step by Step

Summary

Component, API and files

Component

API

- ❑ Initialization
- ❑ Service Functions
- ❑ Callback Functions

C/H Files

- ❑ Component Files
- ❑ Configuration Files
- ❑ Parameter Files

Configuration- and component-dependent functions (API)

Delivered files (generated for CANdesc)

Generated by
- ❑ CANgen
- ❑ GENy

The delivered CANbedded C code components consist of:

Component files:
Files that form the component (C, H), that contain the code of the component and its functionalities. These files can be found in the delivery (except for the files for the CANdesc component. Those are generated completely).

Configuration and parameter files:
These are files that are generated by the configuration too and that tailor the component to provide functionality as you have selected. As these files are generated, do not change them manually. Any change is lost after the next generation process.

On the other hand each component provides a large application interface. Despite this the usage of a component can be seen very simple:

Initialization: there is a function to initialize the component (xxxInitPowerOn).

Service function: There are all functions so use the component (e.g. CanTransmit to send a CAN message).

Callback functions: at predefined states of the component it calls back to the application to get information how to continue. The components do not work without any interaction with the application. This is a very powerful means for the application to control the work flow of the CANbedded software components.

Customer application files ———————————————————

Folders for delivered and ———————————————————
generated Vector files

```
⊟ 📁 Project
  ⊟ 📁 Application
     └ 📁 ApplicationFiles
  ⊟ 📁 Vector
```

Slide: 9

On the developer side the project could look like this:

There is a project folder containing a large amount of application files. And depending on your strategy there could be a folder containing all files delivered from Vector and that are generated by the configuration tool.

9

Introduction

Components and Project Folders

> Delivery

Step by Step

Summary

Slide: 10

A delivery by CD starts automatically when inserting the CD and looks similar like shown above.

Click the link to save the install shield executable on your hard disc. For security reasons the browser normally does not allow to start an executable (.exe) file.

This install shield executable has the name of the delivery followed by _Setup.exe.

Agenda

...Setup.exe

❶ **install**

Components

API
- ❑ Initialization
- ❑ Service Functions
- ❑ Callback functions

Files
- ❑ Component Files

GENy
CANgen

```
Project
├── Application
│   └── ApplicationFiles
├── Vector
│   └── _gentool
│       ├── CANgen
│       └── GENy
│           └── Components
├── _common
├── _doc
├── diag
├── drv
├── nm
├── tpmc
└── furtherComponents
```

e.g.
can_drv.c
can_def.h
il.c
il_def.h

vector

---

1. STEP – Install tool and component files

Start the install shield executable and follow the instructions coming up. Make sure to enter the correct path where files should be installed to. There will be installed the component files and the configuration tool.

In the project folder view the configuration tool will be installed below the folder _gentool.

- ❑ If you use CANgen this will be installed directly below _gentool and the folder is called CANgen.
- ❑ if you use GENy there will be installed two folders, one for the framework (GENy) and one for the component DLLs (Components).

The component files are installed as shown above. there is a _common folder for common files and a _doc folder for all the delivery documentations. The component files are stored in a folder with the name of the component (e.g. diag for diagnostics components, drv for the CAN Driver, nm for Network Management, etc.).

...Setup.exe

Components

API
- Initialization
- Service Functions
- Callback functions

Files
- Component Files

❶ install

❷ configure

DBC

GENy
CANgen

Project
  Application
    ApplicationFiles
  Vector
    _gentool
      CANgen
      GENy
        Components
    _common
    _doc
    diag
    drv
    nm
    tpmc
    furtherComponents
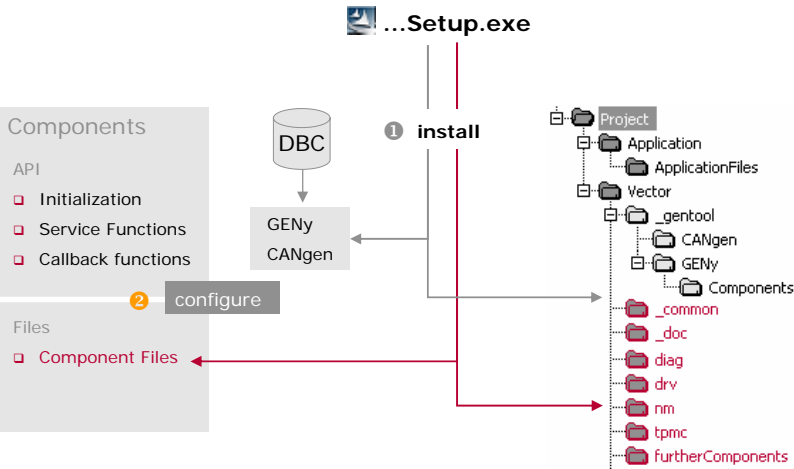
2. STEP – Configure components

Now start the configuration tool (CANgen or GENy – for more information about how to use the tools refer to the Online Help). Configure the components, switch on or of necessary functionality, callback functions, etc.

3. Generate configuration and parameter files

...Setup.exe

Components

API
- Initialization
- Service Functions
- Callback functions

❷ configure

❸ generate

Files
- Component Files
- Configuration Files
- Parameter Files

e.g.
il_par.c
il_par.h
il_cfg.h

DBC

GENy
CANgen

❶ install

Project
  Application
    ApplicationFiles
  Vector
    _gentool
      CANgen
      GENy
        Components
    _common
    _doc
    diag
    drv
    nm
    tpmc
    furtherComponents
    gendata

---
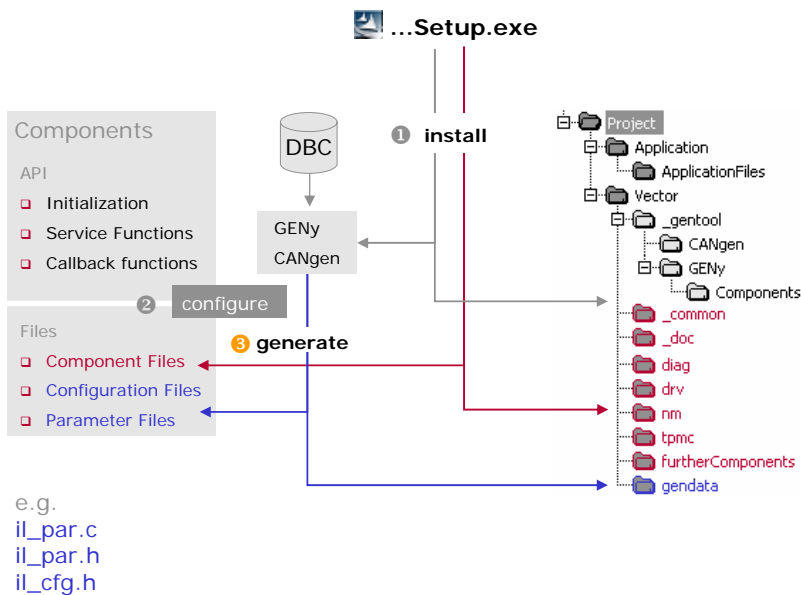
3. STEP – Generate configuration and parameter files

After you have configured the components start the generation process by clicking the flash button of the configuration tool. Make sure you have adjusted the path correctly where the tool generated the files in.

The easiest way would be to use a folder like above, called gendata where all generated data is stored. Then you know that all data in this folder must not be changed by hand.

The files that are generated for configuration and parameters are named like shown above.

*_par – parameter files

*_cfg – configuration files

4. Adapt application files

...Setup.exe

Components

API
- Initialization
- Service Functions
- Callback functions

❷ configure

Files
- Component Files
- Configuration Files
- Parameter Files

DBC

GENy
CANgen

❶ install

❸ generate

Project
- Application
  - ApplicationFiles
- Vector
  - _gentool
    - CANgen
    - GENy
      - Components
  - _common
  - _doc
  - diag
  - drv
  - nm
  - tpmc
  - furtherComponents
  - gendata

❹ adapt

Application files

```
//Include components
#include ...

//Initialize components
CompInitPowerOn();

//Service functions

//Cyclic Component Tasks
TaskXms()
{
CompTask();
}

//Callback functions
```

vector

---

4. STEP – Adapt application files

After the files are generated you have to modify your application to use the functionality of the CANbedded software components.

There are the C typical points to be regarded when using functionality of a component.

Include:
Your software must know where to find the functions you want to use. Therefore you have to include the necessary header files of the components to be used.

Initialization:
Before you can use a component it must be initialized.

Service functions:
To use the actual functionality of the component call service functions like e.g. CanTransmit.

Cyclic component tasks
A very special sort of service functions are the cyclic component tasks. Only the CAN Driver is event triggered (when not used in polling mode). All other components need a task function to be called cyclically within a predefined time. This call cycle is the base time for internal time dependent functionality like e.g. timeout monitoring.

Callback functions:
interact with the component to control component's activities.

Some examples for include, initialization, cyclic cyclic component task and service and callback functions

Application files

```
//Access to components
#include fileA
#include fileB      #include ccl_inc.h //only one initialization
                    function when using CCL
//Initialize components, interrupts disabled
CanInitPowerOn();
NmInitPowerOn();    CclInitPowerOn(); //only one
..                  initialization function when using CCL
enable Interrupts


//Service functions              //Callback functions
IlRxStart();                      ApplCanMessageReceived()
IlTxStart();                      {
                                     //application code
                                  }
//Cyclic Component Tasks
TaskXms()
{
  CompATask();
}
                    CclScheduleTask(); //one function call
TaskYms()           for all component task functions
{
  CompBTask();
}
```

vector

---

4. STEP – Adapt application files

Include:
There is an order that has to be kept when including the files for CANbedded software components. If you use the CCL component there is only one include left, the one of ccl_inc.h.

Initialize components with disabled interrupts:
The CanInitPowerOn() must be the first initialization followed by the higher layer components like Network Management or Transport Protocol, etc. If you use the CCL component, there is only one call of CclinitPowerOn() necessary.

Cyclic component tasks:
In the configuration tool you have to enter the cycle time (normally in ms) in which you call the component's task functions (CompATask( ), CompBTask( ), etc.). This information is the basis for the component to calculate its internal timing. So you have to call the function in the given cycle time. This can be done form a operating system task or a timer. If you use the CCL component just call CclScheduleTask() within a suitable cycle time and the single calls will be performed by the CCL (CCL provides another method for the cyclic calls, refer to the documentation of the CCL component for more information).
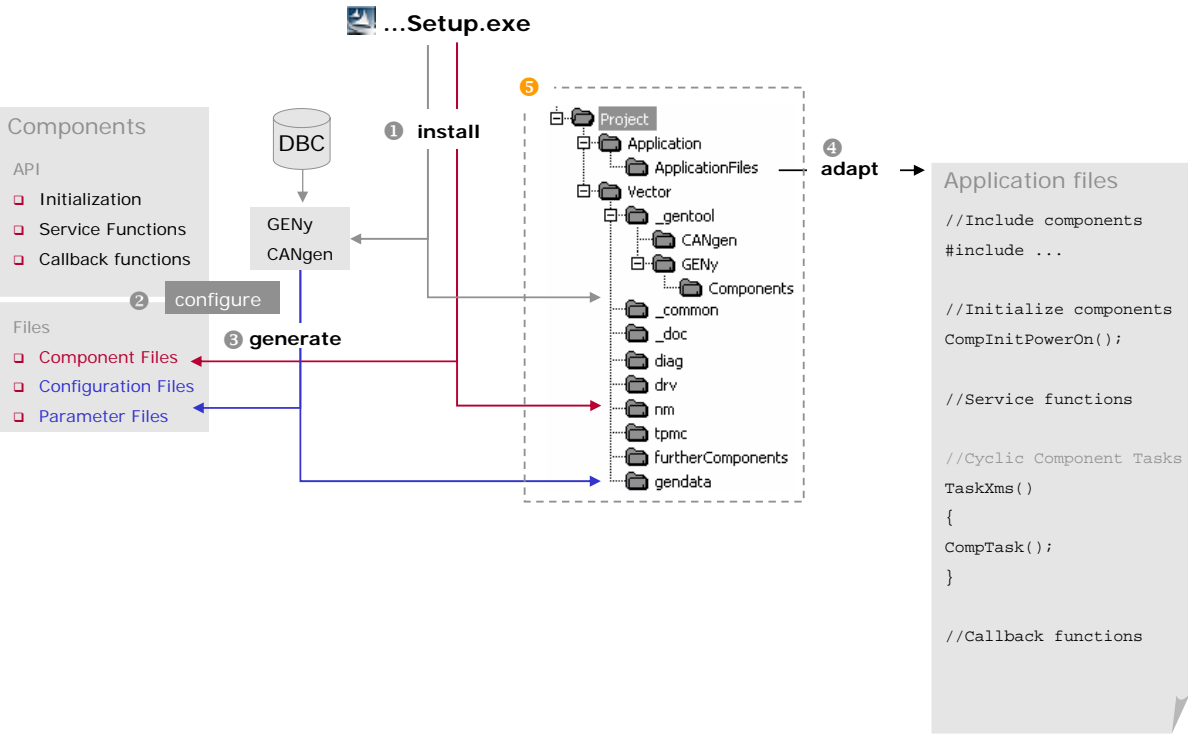
Service functions:
Service functions are e.g. CanTransmit to transmit a CAN message, IlRxStart to start the transmitting path of the Interaction Layer, etc. These are functions to use the components.

Callback functions:
e.g. ApplCanMessageReceived. Callback functions are always marked with Appl... in the beginning of the name. The activity of the component is halted and the application is asked to meet a decision how to continue. Often the return code of the function is used for this decision, but now always. In some cases the application has to do some other action.
Tip: for the beginning provide all necessary callback functions but leave them empty just to prevent linker errors. Then start to fill them with code one after the other.

...Setup.exe

Components

API
- Initialization
- Service Functions
- Callback functions

❷ configure

Files
- Component Files
- Configuration Files
- Parameter Files

DBC

❶ install

GENy
CANgen

❸ generate

❺

- Project
  - Application
    - ApplicationFiles
  - Vector
    - _gentool
      - CANgen
      - GENy
        - Components
    - _common
    - _doc
    - diag
    - drv
    - nm
    - tpmc
    - furtherComponents
    - gendata

❹ adapt →

Application files

```
//Include components
#include ...

//Initialize components
CompInitPowerOn();

//Service functions

//Cyclic Component Tasks
TaskXms()
{
CompTask();
}

//Callback functions
```

5. STEP – Add files in your project or makefile

Now the component files are configured (generated files) and your application is adapted to use the components and their functionality. What is left is to add the new files (component files and generated files) to your compiler or makefile.
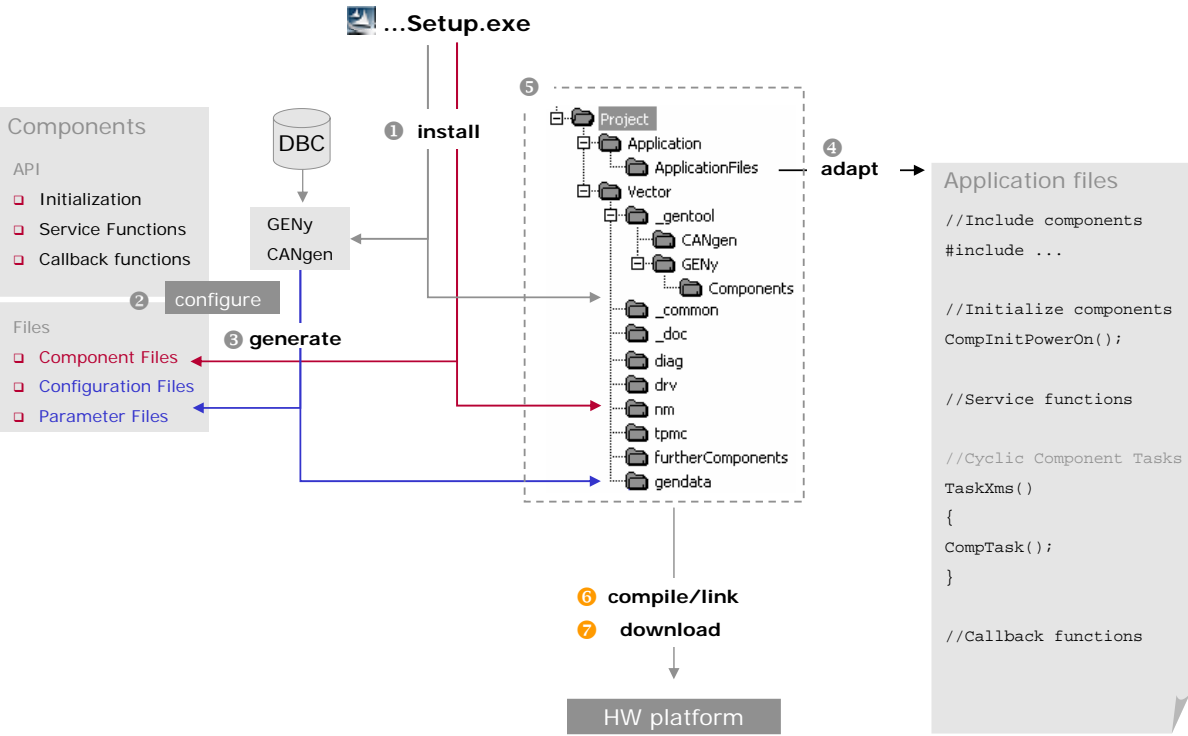
6. STEP – Compile and link your project with CANbedded software components

Now start the compiler for a first compile and link process. Make sure you have provided all necessary callback functions.

7. STEP – Download the executable to your hardware and test

The first target should be a basically working CAN communication. Upon this you can build all further development. So download your executable and watch the bus for communication. Use perhaps CANoe or CANalyzer (products of Vector Informatik GmbH). If you see the CAN messages on the bus and no error frames, the first target is reached.

Agenda

Introduction

Components and Project Folders

Delivery

Step by Step

> Summary

Step by Step

❶ .exe **Setup.exe** – Install Configuration Tool (CANgen or GENy) and CANbedded Components

❷ .gny
.ccf **Configure** Components with Configuration Tool and data base file

❸ _par.h
_cfg.h **Generate** Parameter and Configuration Files

❹ appl.c **Adapt** your application to utilize CANbedded Components

❺ .mak **Add** CANbedded to your Project or makefile

❻ .hex **Compile** and **Link** the Project (application with CANbedded)

❼ .hex **Download** to target and test

vector

Use the list above as a short recipe for using the CANbedded software components out of the delivery.

Summary

Documentation

- Beginners of CANbedded components should read the UserManual_Startup_<OEM>_CANbedded first (if available).

- If not part of the delivery read the component specific user manuals UserManual_<Component>. And if you use the CCL read the CCL user manual first.

- There is a set of user manuals and references for every component as you see below.

———————————————————— Document Types ————————————————————

User Manual	Technical Reference(SW)	Technical Reference(HW)
First steps to get an example executable running for each component or the complete delivery (if available for OEM). UserManual_CCL, UserManual_Startup_<OEM>_ CANbedded, …	More detailed information about the component, API…	Hardware specific information of the component if available.
	TechnicalReference_candrv, …	TechnicalReference_CAN_HC12, …

vector

Thank you for your attention.

For detailed information about Vector

and our products please have a look at:

www.vector-informatik.com

Author:

Klaus Emmert

Vector Informatik GmbH

Ingersheimer Str. 24

70499 Stuttgart

vector