Manual

Version 2.7

Vector Informatik GmbH, Ingersheimer Str. 24, D-70499 Stuttgart
Tel. +49 711 80670-0, Fax +49 711 80670 111,
Email can@vector-informatik.de, Internet http://www.vector-informatik.de

Subsidiaries

France

Vector France SAS

168, Boulevard Camélinat
F-92240 Malakoff

Tel.: +33 1 4231 4000
Fax: +33 1 4231 4009

http://www.vector-france.com

Japan

Vector Japan Co., Ltd.

Nishikawa Bldg. 2F
3-3-9 Nihonbashi, Chuo-ku
J-103-0027 Tokyo

Tel.: +81 3 3516 7850
Fax: +81 3 3516 7855

http://www.vector-japan.co.jp

Sweden

VecScan AB

Fabriksgatan 7
S-41250 Göteborg

Tel.: +46 031 79901 35
Fax: +46 031 79903 05

http://www.vecscan.com/

USA

Vector CANtech, Inc.

Suite 550
39500 Orchard Hill Place
USA-Novi, Mi 48375

Tel.: +1 248 449 9290
Fax: +1 248 449 9704

http://www.vector-cantech.com

The addresses of our distributors are available on our website:

http://www.vector-informatik.com

International Quality Standard Certificate

The Quality Management of Vector Informatik GmbH is continuously certificated since 1998-08-19:

- 2001-11-27
  acc. to DIN EN ISO 9001:2000-12
  Certificate number: 70 100 1498
- 1998-08-19
  acc. to DIN EN ISO 9001:1994-08
  Certificate number: 70 100 F 1498 TMS

Typographic Conventions

Note:	Identifies important notes
•	Identifies enumerations (bullet items)
➔ '1.0 Introduction'	Identifies references to further chapters of this manual
[OK]	Notation for buttons in dialogs
<TAB>	Notation for keys on the computer keyboard
<Ctrl>+<Z>	Notation for keys of the computer keyboard which should be pressed simultaneously
Add… **File|File open…**	Notation for menu, command and dialog names
`on message 0x100`	Notation for MS-DOS syntax or program code
✗ *Start...*	Identifies problems to be worked on and solved with the help of the tutorial

Contents

Looking for other topics or entries? Online Help contains further information.

Looking for other topics or entries? Online Help contains further information.

1 Introduction

1.1 Overview of CANdb++

All communication-relevant data that are processed in a networked CAN bus system as well as their interrelationships are usually administered in a central communications database.

CANdb++ is a data administration program with which these communication databases can be created and modified in the form of CAN databases.

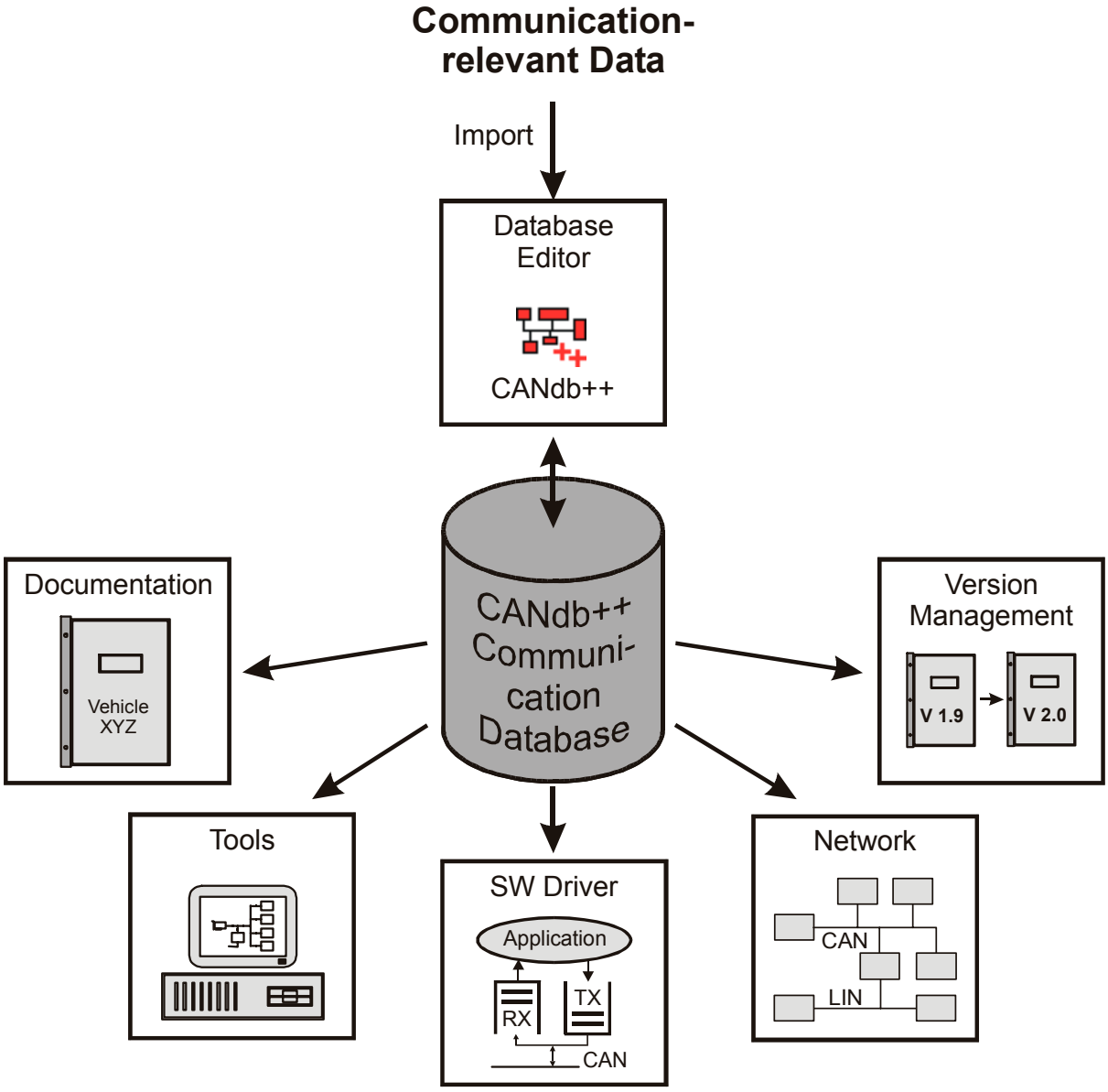

Figure 1-1: CANdb++ as central tool for managing communication data

1.1.1 CANdb++ Program Versions

CANdb++ is available in the following program versions:

- CANdb++
- CANdb++ Admin

Please refer to Table 1-1 for the functional features of the program versions.

1.1.2 CANdb++ Functional Features

Function	Program Version	
	CANdb++	CANdb++ Admin
Creating and modifying CANdb++ databases (*.mdc)		•
Creating and modifying CANdb network files (*.dbc)	•	•
Creating and modifying user-defined attributes	•	•
Creating and modifying value tables	•	•
Showing the communication matrix	•	•
Create variants of an object	•	•
Consistency check of a CAN database	•	•
Creating and modifying objects of the "Vehicles" object type		•
Comparing objects	•	•
Comparing CAN databases Export of differences in text files (*.csv)		•
Import objects and attributes		•
Export objects and CAN databases	•	•
Version administration for CAN databases and objects		•
Generate reports		•
Timing analysis (estimation of bus load)		•

Table 1-1: Functional Features of CANdb++ Program Versions
(• Function available)

Note:	To achieve Version Administration capabilities with CANdb++ Admin it is necessary to have Microsoft® Visual Source Safe installed. Moreover, all of the requirements named in chapter '3.1 Preconditions for Version Administration Capability' (➔ Page 25) must also be satisfied.

1.1.3 Integration in the Vector CAN-Tools Toolbox

CANdb++ is a central tool in the Vector CAN-Tools toolbox, and it can be started directly from Vector CAN tools such as CANalyzer, CANoe, CANape and CANscope. This provides the Vector CAN tools with direct access to communication-relevant data via CANdb++.

Communication-relevant data are defined, modified and managed entirely within CANdb++; the Vector CAN tools only read-access these data.

The communication-relevant data must exist in the form of CANdb network files (*.dbc) so that the Vector CAN tools can access them.

To also permit the use of data from CANdb++ databases (*.mdc) CANdb++ provides a data export option. CANdb++ databases can be exported to one or more CANdb network files (*.dbc). In this way the user can either export the data of an entire CANdb++ database or only those relevant to a Vehicle, Network or Control Unit.

1.2 CANdb++ Data Model

Various object types are available in the Overview Window for modeling the communications structure of a vehicle's CAN bus system.

Object type of the object to be linked	**Object type of the object to which a link can be made**						
	Vehicles	Networks	Control Units	Node Groups	Network Nodes	Messages	Signal Groups
Networks	•						
Control Units (ECUs)	•						
Environment Variables			•				
Node Groups		•					
Network Nodes		•	•	•			
Messages					•		
Message Signals					•		•
Signal Groups						•	
Signals						•	

Table 1-2: Possible Links

Note:	A **Gateway** is a special Control Unit, that is used as a connecting element between two or more CAN buses. For detailed information on the individual object types please refer to the online Help glossary.

Adding a link establishes a connection (Relation) between two objects of different object types. By linking a signal with a message, for example, the user can define the message in which this signal should be transmitted.

Possible links are shown in Table 1-2.

1.3 User Requirements

The user should have a basic working knowledge of Microsoft® Windows® in order to work with CANdb++ or CANdb++ Admin.

You can obtain information on working with Microsoft® Windows® from the Microsoft® Windows® manuals, Microsoft® Windows® online Help or additional literature.

1.4 Software Installation

1.4.1 Hardware Requirements

To install and run CANdb++ your hardware must satisfy the following requirements:

- IBM-compatible PC
- Processor: At least Pentium
- Working memory (RAM): At least 32 MB
- Operating system: Microsoft® Windows® 95, 98, NT or 2000

1.4.2 Installation

Note:	You must have administrator rights to install the software! The user can switch the language of the menus and dialogs at any time after installation (➔ '4.2.1 VECTOR.INI', Page 28).

Proceed as follows to install CANdb++:

1. Insert the installation CD in your CD drive.
2. Call the installation program SETUP.EXE.
3. Follow the instructions of the installation program.

1.5 CANdb++ Program Window

The following elements are arranged in the CANdb++ program window:

- Title bar
- Menu bar
- Toolbar
- Version Administration Toolbar
  (The Version Administration Toolbar is only available in the CANdb++ Admin program version!)
- Working area with various windows
- Status bar

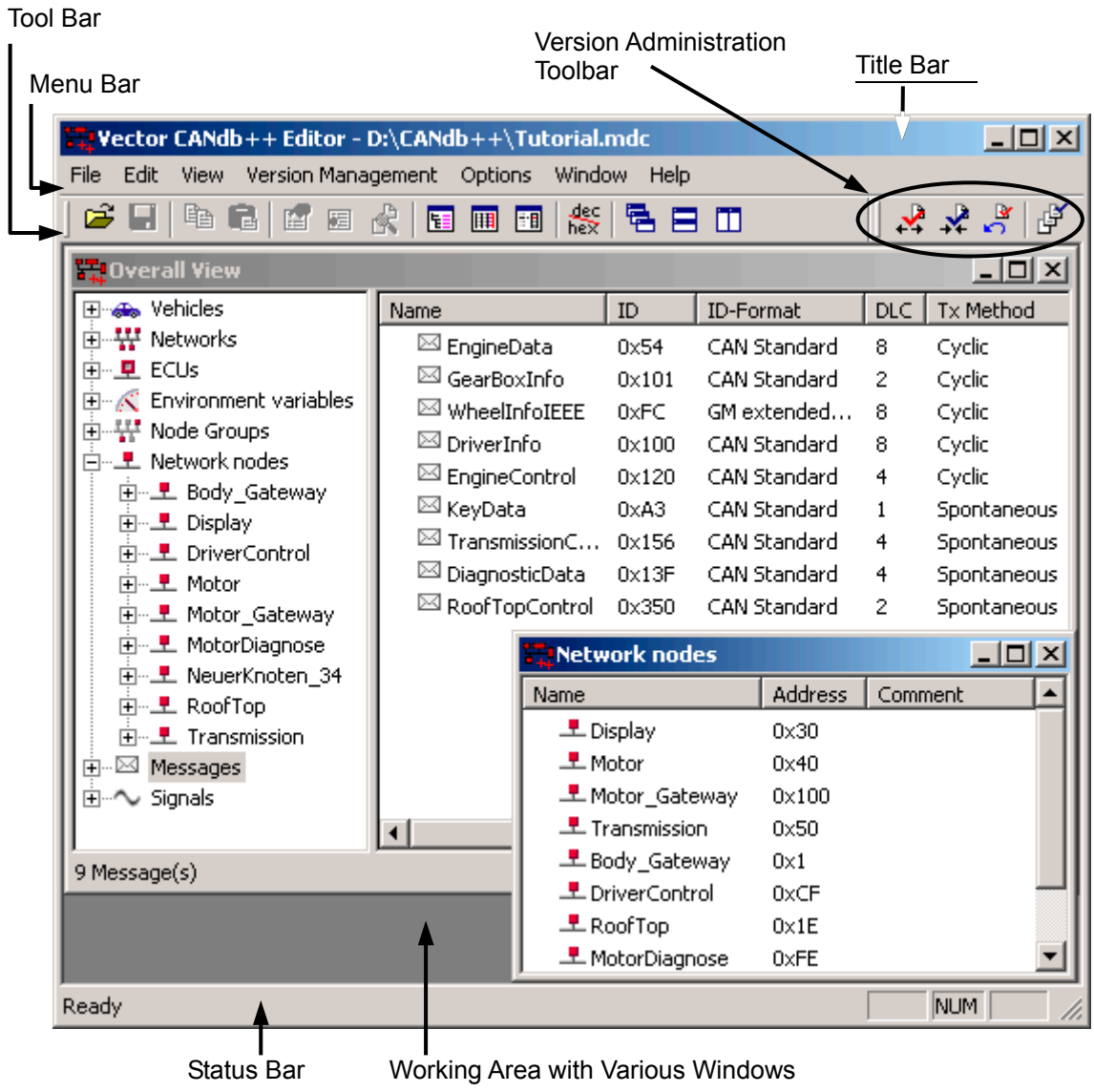

Figure 1-2: CANdb++ Program Window

1.6 CANdb++ Online Help

CANdb++ and CANdb++ Admin provide a comprehensive online Help function that can be called by pressing the **[Help]** button or the <F1> key.

Note:	If you choose the command **Using Help** (**Help** menu) or press the <F1> key while CANdb++ online Help is active, you get information on using and configuring the online Help function.

2 Tutorial

The purpose of this tutorial is to familiarize you with the CANdb++ user interface concept and the most important of the CANdb++ functions.

At the beginning of each chapter are problems that you can solve with the help of the explanations that follow. These problems are identified by a ✗ symbol at the beginning of the line and by italic font style.

When creating a new CAN database the following steps are performed in the sequence shown below:

1. Starting the program (➔ Page 7).
2. Setting up a new CAN database (➔ Page 8).
3. Creating and modifying the necessary objects.
4. Creating new objects (➔ Page 9).
5. Copying existing objects (➔ Page 10).
6. Modifying existing objects (➔ Page 11).
7. Linking the objects (➔ Page 13).
8. Displaying the communications matrix (➔ Page 15).
9. Creating (➔ Page 16) and assigning (➔ Page 18) value tables.
10. Creating user-defined attributes (➔ Page 19) and modifying values of the user-defined attributes of individual objects (➔ Page 21).
11. Creating object variants (➔ Page 22).
12. Performing the consistency check (➔ Page 23) and making necessary corrections to the CAN database.

2.1 Program Start

✗ *Start CANdb++.*

CANdb++ can be started as follows:

- Double click the program icon  in the CANdb++ program group
- Double click CANDB.EXE (e.g. in Microsoft® Explorer)
- Click the program name "CANdb++" in the appropriate sub-folder of the Microsoft® Windows® Start menu.

After starting CANdb++ the CANdb++ program window appears. The working area is still empty, i.e. it does not contain any windows.

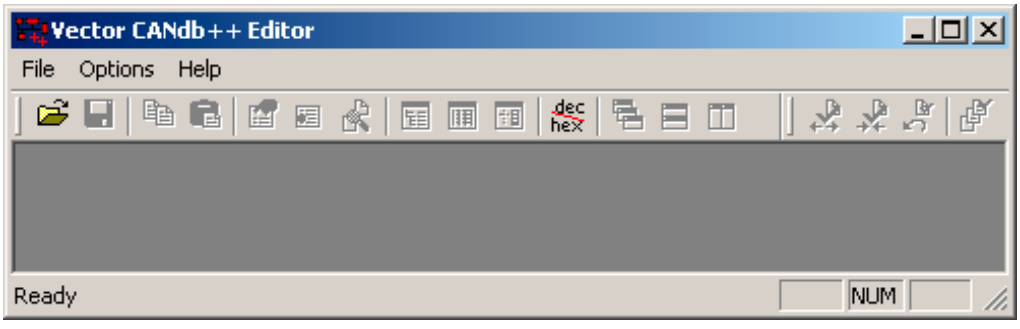

Figure 2-3: CANdb++ Program Window after Program Start

2.2 Creating a New CAN Database

✗ In the Data directory of your CANdb++ installation create a new CANdb++ database with the name Tutorial.mdc or Tutorial.dbc.

With CANdb++ you can set up CAN databases of the following types:

- CANdb++ databases (*.mdc) (only CANdb++ Admin)
- CANdb network files (*.dbc)

Proceed as follows to create a new CAN database:

- Choose the **Create Database** command (**File** menu).
  First, the **Template** dialog is opened.
- There you decide whether to chose a template for a CANdb Network (.dbc) or for a CANdb++ Database (.mdc).
  Next you can select one of the available templates by a doubleclick or by a single click and chosing **[OK]**.
- After you have selected a template, the **New Database File** dialog is opened, in which the memory location, file name and file type can be defined for the CAN database to be created.

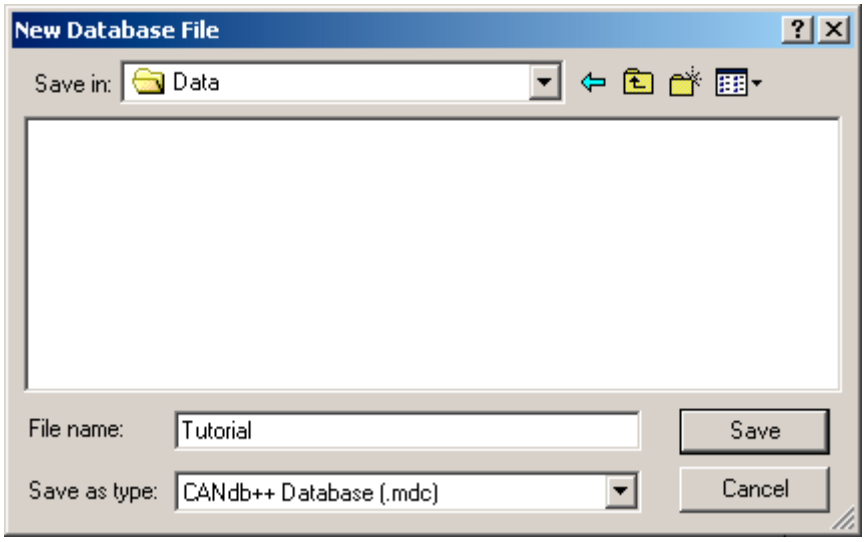

Figure 2-4: **New Database File** dialog

- Select the directory in which you wish to save the new CAN database (Data directory of your CANdb++ installation or CANdb network). Select the desired file type (e.g. CANdb++ database) and enter the desired file name (e.g. Tutorial). It is not necessary to input a file name extension - it will be assigned automatically by CANdb++ according to the selected file type.
- Press the **[Save]** button.
- The new CAN database (e.g. Tutorial.mdc or Tutorial.dbc) is set up and is shown in the Overview Window. Shown on the left side of the Overview Window is a hierarchical overview of the available object types.

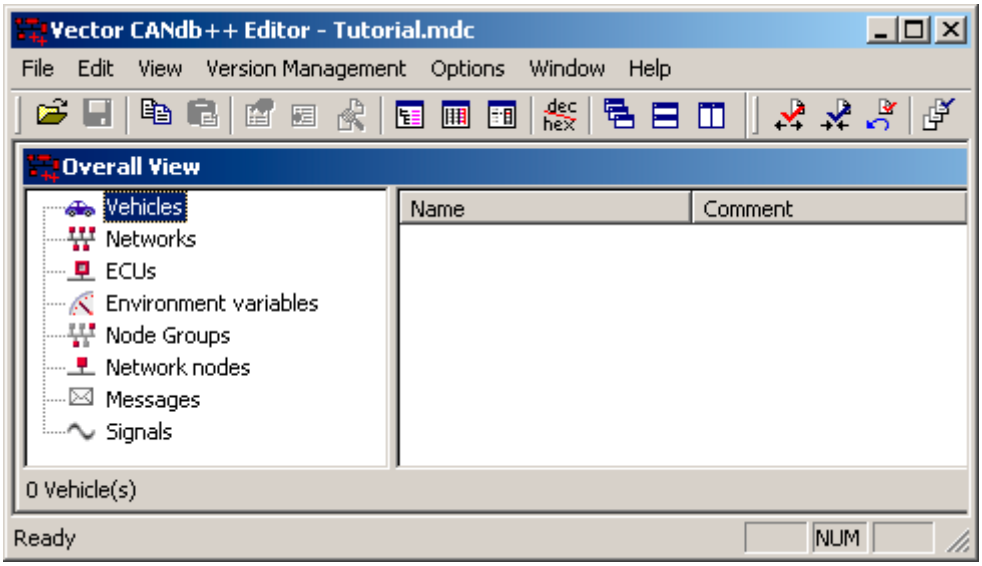

Figure 2-5: New CANdb++ Database in the Overview Window

2.3 Creating and Modifying Objects

2.3.1 Creating New Objects

✗ *Create the following objects in the CAN database Tutorial.mdc or Tutorial.dbc:*

- *Vehicles (Coupe) (only CANdb++ Admin)*
- *Networks (Body, PowerTrain) (only CANdb++ Admin)*
- *Control units (Combi, DriverControl, ECU_Motor (only CANdb++ Admin))*
- *Environment variables (EnvTemp)*
- *Network nodes (Body_Gateway, Display, Motor, Motor_Gateway, SteeringLock)*
- *Messages (DriverInfo, EngineControl, KeyData, TransmissionData)*
- *Signals (DisplayTemp, GearSelect, RunMode, Information)*

Proceed as follows to create a new object, e.g. a signal, in the Overview Window:

- Select the object type for which you wish to set up a new object. To do this click the object type on the left side of the Overview Window. The selected object type gets a color background (Default: Blue). Appearing on the right side of the Overview Window are all objects of this type that have already been created. Using this method you would select, for example, the Signals object type if you wish to create a new signal.
- Choose the **New** command (**Edit** menu). This opens an object dialog. Appearing in the input boxes of the object dialog are suggestions for concrete values of system parameters.
  For example, when you have created a new signal the **Signal…** object dialog appears.

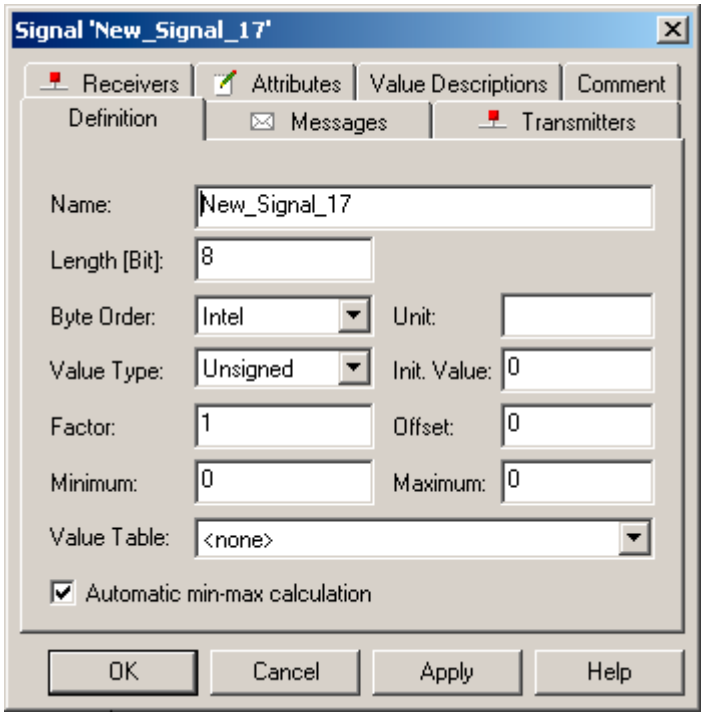

Figure 2-6: Object Dialog for a New Signal

- Change the values of system parameters in the object dialog and press the **[OK]** button. The newly created object appears in the Overview Window. Appearing in the table columns on the right side of the Overview Window are the values of the object's system parameters.

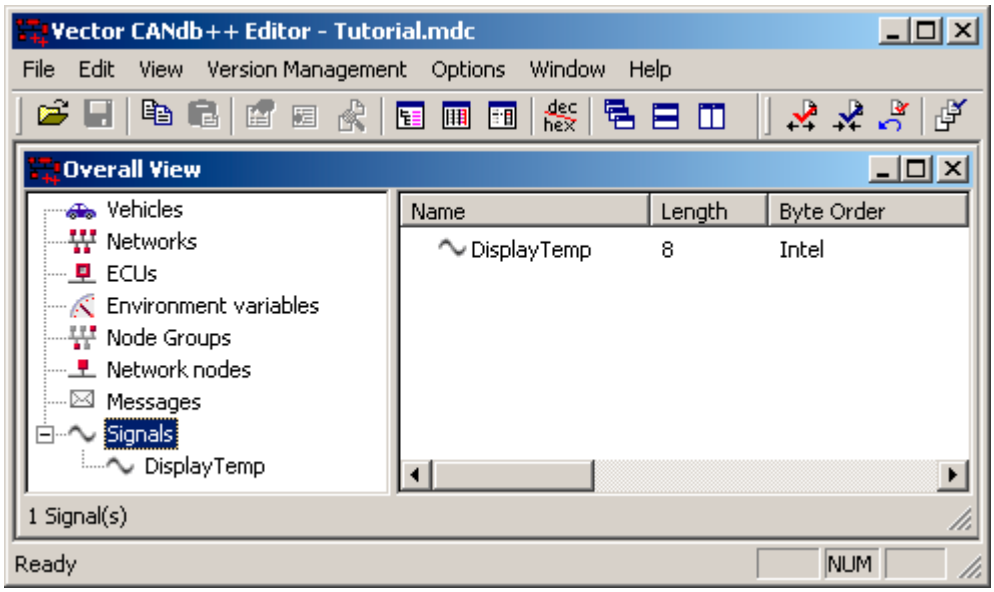

Figure 2-7: Overview Window with a Signal

2.3.2 Copying Existing Objects

Create a copy of the signal RunMode in the CAN database Tutorial.mdc or Tutorial.dbc and give it the name OperateMode.

Proceed as follows to create a copy of an existing object, e.g. of a signal:

1. Select the object that you wish to copy. To do this, click the object in the Overview Window. The selected object has a color background (Default: Blue).
2. Copy the selected object with the **Copy** command (**Edit** menu) to the Clipboard.
3. Create a copy of the object located in the Clipboard using the **Paste** command (**Edit** menu).
4. The copy of the object appears in the Overview Window.
5. Modify the copy of the object (➔ Page 11).

Note:	After creating a copy of an object this copy is completely independent of the original object! Changes to the original object are not mirrored in the copy, and if the changes are desired in the copy they must be made manually afterwards.

2.3.3 Modifying Existing Objects

✗ *Modify the network nodes in the CAN database Tutorial.mdc or Tutorial.dbc such that each node has a different address.*

2.3.3.1 Modifying an Object's Parameters in the Object Dialog

Proceed as follows to modify the parameters of an object:

1. Select the object to be modified. To do this, click the object in the Overview Window. The selected object gets a color background (Default: Blue).
   In this manner you would select, for example, the network node Display.
2. Choose the **Edit** command (**Edit** menu). This opens an object dialog. Appearing in the input boxes of the object dialog are the concrete values for system parameters.
   For example, when you select the network node Display the **Node 'Display'** object dialog appears.

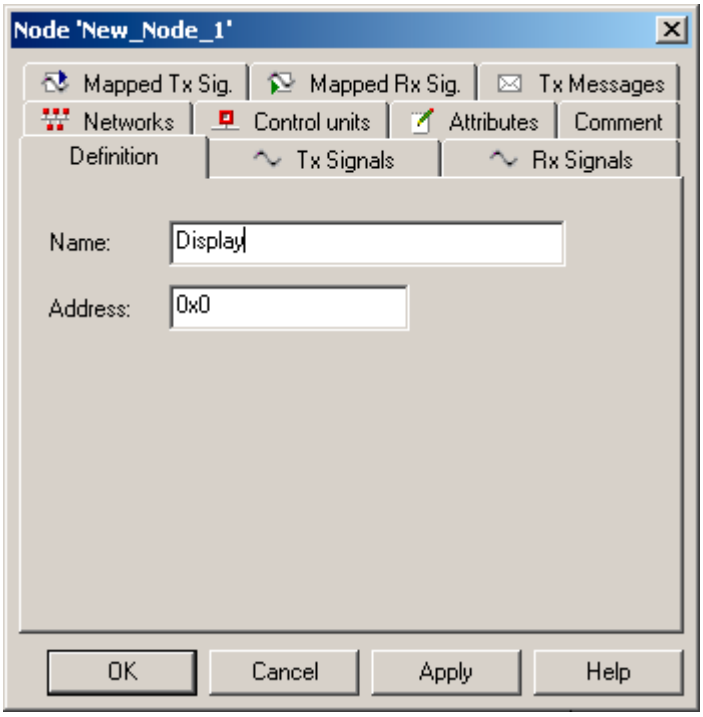

Figure 2-8: Object Dialog of the Network Node Display

3. Modify the system parameters according to your requirements.
4. Press the **[OK]** button. The changed system parameters of the object appear in the table on the right side of the Overview Window.

2.3.3.2 Modifying an Object's Parameters in the Table in the Overview or List Window

Proceed as follows to modify an object's parameters directly in the table in the Overview Window or List Window:

1. Click the table cell containing the parameter of the object to be modified. The corresponding table line gets a color background (Default: Blue). The activated cell is shaded with a different color (Default: Light gray).
2. Activate the editing mode for the activated cell by pressing the <F2> key or by clicking the cell again. The cell gets a frame and a color background (Default: Blue), and if applicable a list of possible parameters is shown.
3. Change the parameter by entering a different value or selecting a different option from the list that is shown.
4. Press the <Esc> key if you wish to abort the procedure, i.e. you wish to leave the parameter value unchanged.
   Press the < ↵ Return> key if you wish to accept the changed value.

Note:	You can switch the selection of the active cell within the table using the arrow cursor keys <←>, <→>, <↑> and <↓>.

2.4 Linking Objects

✗ *Link the following objects in the CAN database Tutorial.mdc or Tutorial.dbc:*

- *Signals with messages (GearSelect-TransmissionData, DisplayTemp-DriverInfo, Information-DriverInfo, OperateMode-EngineControl, RunMode-KeyData)*
- *Messages with network nodes (TransmissionData-Display, EngineControl-Motor_Gateway, KeyData-SteeringLock, DriverInfo-Body_Gateway)*
- *Message signals with network nodes (DisplayTemp-Display, GearSelect-Body_Gateway, OperateMode-Motor, RunMode-Body_Gateway)*
- *Network nodes with node groups (Motor-PowerTrain_Base, Motor_Gateway-PowerTrain_Base) (only CANdb++ Admin)*
- *Network nodes with control units (Body_Gateway-Combi, Display-Combi, Motor-ECU_Motor, Motor_Gateway-Combi, SteeringLock-DriverControl) (only CANdb++ Admin)*
- *Network nodes with networks (Body_Gateway-Body, SteeringLock-Body, Display-Body, Motor-PowerTrain, Motor_Gateway-PowerTrain) (only CANdb++ Admin)*
- *Node groups with networks (PowerTrain_Base-PowerTrain) (only CANdb++ Admin)*
- *Control units with vehicles (Combi-Coupe, DriverControl-Coupe, ECU_Motor-Coupe) (only CANdb++ Admin)*
- *Networks with vehicles (Body-Coupe, PowerTrain-Coupe) (only CANdb++ Admin)*

By linking two objects of different object types you can establish a connection (Relation) between these two objects. For example, by linking a signal with a message you can define the message in which this signal should be transmitted.

2.4.1 Linking by 'Drag and Drop'

Proceed as follows to link two objects by 'Drag and Drop':

1. On the left side of the Overview Window show the structure level that contains the object with which a link should be made. To do this, click the ⊞ symbol in front of the object type for this object. The subordinate structure level is then shown.
   For example, if you wish to link a signal to a message click the ⊞ symbol in front of the Messages object type.
2. Open a List Window for the object type of the object you wish to link. To do this, choose the **List-…** command (**View** menu) for the relevant object type.
   For example, if you wish to link a signal to a message choose the **List-Signals** command to open a List Window with signals.
3. If necessary change the size of the Overview Window and the List Window so that the relevant areas are visible in both windows.

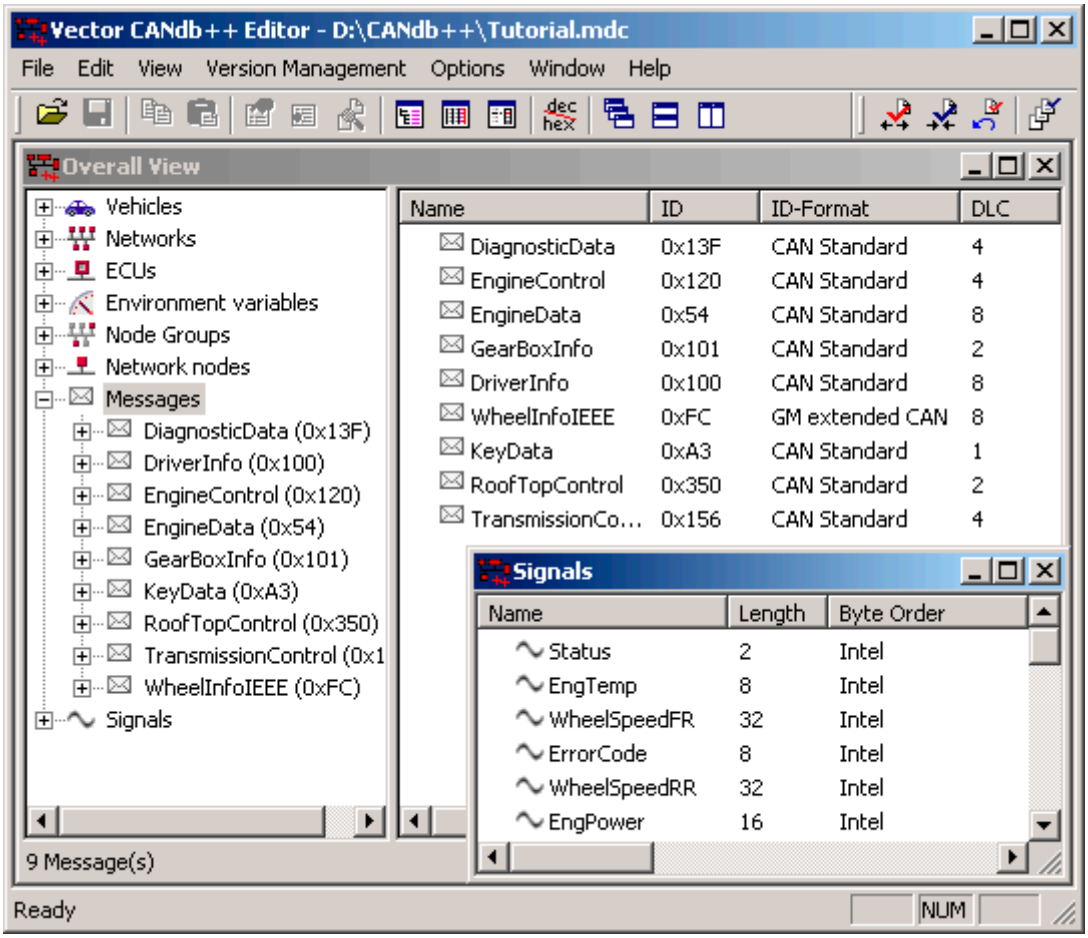

Figure 2-9: Overview Window and List Window with Signals

4. In the List Window select the object that you wish to link.
5. Link the two objects. To do this, drag the selected object from the List Window to the Overview Window while holding down the mouse button. Do not release the mouse button until the mouse pointer is located above the object in the Overview Window with which the link is to be made.

	When the mouse pointer assumes this shape the objects can be linked.
	When the mouse pointer assumes this shape the objects **cannot** be linked.

The link is automatically added after releasing the mouse button. The linked object appears in the Overview Window below the object with which it was linked. The object symbol of the linked object is identified by the link symbol .

For example, the object symbol of a linked signal, i.e. a message signal would appear as follows: 

2.4.2 Linking by Menu Commands

As an alternative to this you could link objects by menu commands as described below:

1. In the Overview Window select the object that you wish to link.
2. Copy the selected object to the Clipboard with the **Copy** command (**Edit** menu).
3. In the Overview Window select the object you would like to link to the object just copied.
4. Add the new link with the **Add link** command (**Edit** menu). The link is then added.

2.5 Showing the Communications Matrix

✗ *Open the Communications Matrix Window that contains the communications matrix of the CAN database Tutorial.mdc or Tutorial.dbc.*

A communications matrix shows the communications relationships between signals, messages and network nodes in table format.

Proceed as follows to have the communications matrix displayed:

Choose the **Communications matrix** command (**View** menu). This automatically opens the Communications Matrix Window with the communications matrix of the active CAN database.

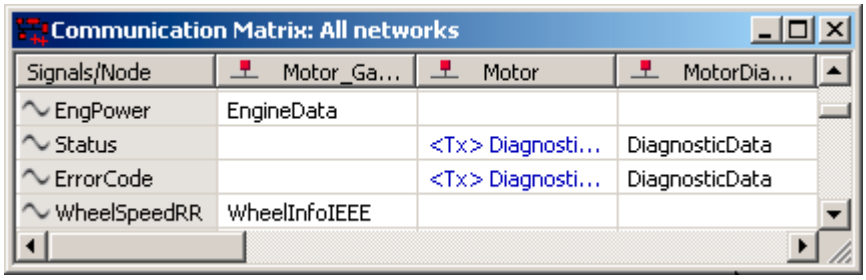

Figure 2-10: Communications Matrix Window

In the Communications Matrix Window the signals are arranged by lines and the network nodes by columns. The table boxes with the signal names have gray shading. The messages shown in the remaining table boxes are the messages in which the signal is transmitted or received by the particular network node.
Transmitted messages are identified as follows:

- Message name in blue font
- "<TX>" in front of the message name

Note:	When the Communications Matrix Window remains open it is automatically updated with each modification to the CAN database.

2.6 Value Tables

2.6.1 Creating Value Tables

In the CAN database Tutorial.mdc or Tutorial.dbc create the value table Colors which assigns the concrete values 0, 1 and 2 to the symbolic identifiers 'red', 'yellow' and 'green'.

In a value table individual signal values or environment variable values can be assigned to symbolic identifiers.

Proceed as follows to create a value table:

1. Choose the **Value Tables** command (**View** menu). The Value Tables Window is opened.

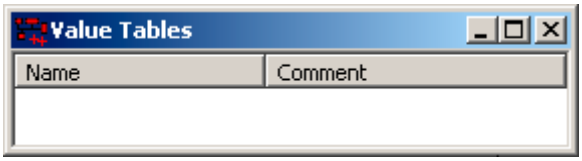

Figure 2-11: Value Tables Window

2. Choose the **New** command (**Edit** menu). The **Value Table** object dialog is opened.

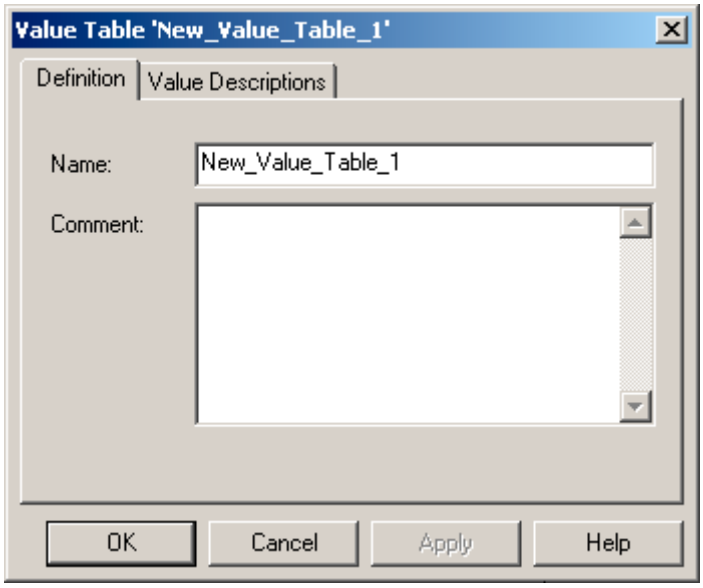

Figure 2-12: Object Dialog of a New Value Table

3. Change the name of the value table. Enter the new name in the **Name** input box.

4. Activate the **Value Description** page by clicking its tab.

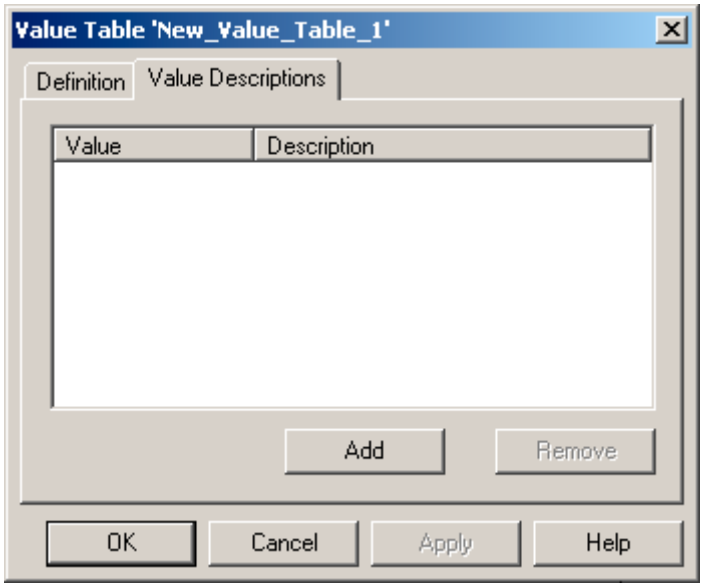

Figure 2-13: **Value Description** Page (**Value Table** dialog)

5. Press the **[Add]** button. A new line is added to the table. Appearing in the **Value** column is a suggested concrete value for the signal or environment variable to which a symbolic identifier should be assigned. Appearing in the **Description** column is a suggested text for this symbolic identifier.
6. Click the cell whose text you wish to change. The cell is selected by a frame and can now be edited.
7. Change the values and symbolic identifiers.
8. Press the <Esc> key if you wish to abort the procedure, i.e. you wish to leave the value or symbolic identifier unchanged.
   Press the < ↵ Return> key if you wish to accept the changed value or changed identifier.
9. Press the **[OK]** button. The newly created value table appears in the Value Tables Window.

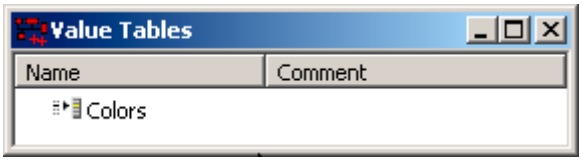

Figure 2-14: Value Tables Window with a Value Table

2.6.2 Assigning Value Tables

In the CAN database Tutorial.mdc or Tutorial.dbc assign the Colors value table to the Information signal.

The value table must be assigned to a signal or environment variable so that the symbolic identifiers that are assigned to individual values in a value table will indeed be assigned to the signal or environment variable values.

Proceed as follows to assign a value table to a signal or environment variable.

1. Select the signal or environment variable to which the value table should be assigned. To do this, click the specific object in the Overview Window or List Window.
2. Choose the **Edit** command (**Edit** menu). This opens an object dialog. For example, if you select the Information signal the **Signal 'Information'** object dialog appears.
3. In the **Value Table** list box select the value table that you wish to assign to the signal or environment variable.

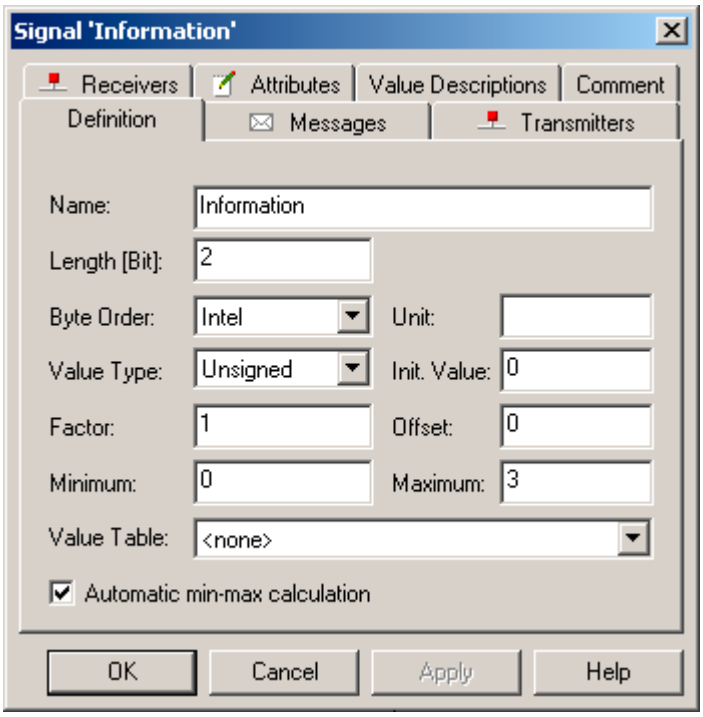

Figure 2-15: Object Dialog for the Signal 'Information'

4. Press the **[OK]** button.

2.7 User-Defined Attributes

2.7.1 Creating User-Defined Attributes

In the CAN database Tutorial.mdc or Tutorial.dbc create the user-defined attribute Release for Vehicles. Concrete values available for the Release attribute should be: 'pending', 'confirmed' and 'denied'.

In addition to system parameters that must be defined when creating an object, objects can also be assigned user-defined attributes.

Proceed as follows to create a user-defined attribute:

1. Choose the **Attribute Definitions** command (**View** menu). This opens the Attribute Definitions Window.

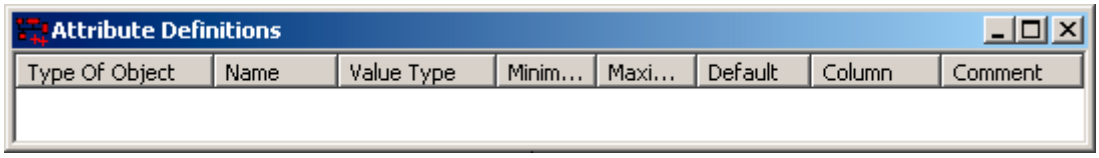

Figure 2-16: Attribute Definitions Window

2. Choose the **New** command (**Edit** menu). The **Attribute Definition** object dialog is opened. Appearing in the input boxes are suggested concrete values for parameters of the user-defined attribute.

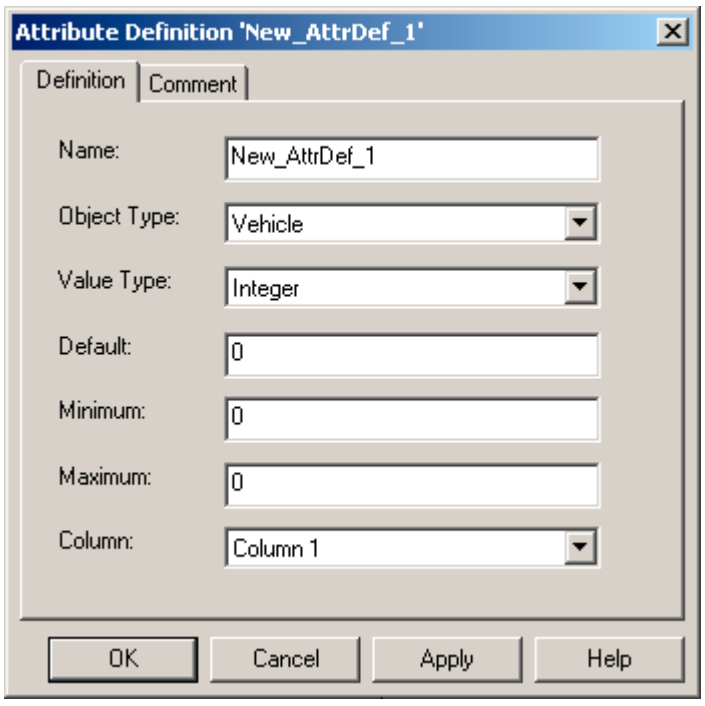

Figure 2-17: Object Dialog for a New User-Defined Attribute

3. Change the parameters of the user-defined attribute. Enter the new name and select the object type and value type.
   To provide concrete texts as values you must select the 'Enumeration' value type. These texts can then be entered in the **Value Range** input box that then appears.
   The value in the **Default** input box is assigned as the default value of the user-defined attribute for each object of the selected object type.
4. Press the **[OK]** button. The newly created user-defined attribute now appears in the Attribute Definitions Window.

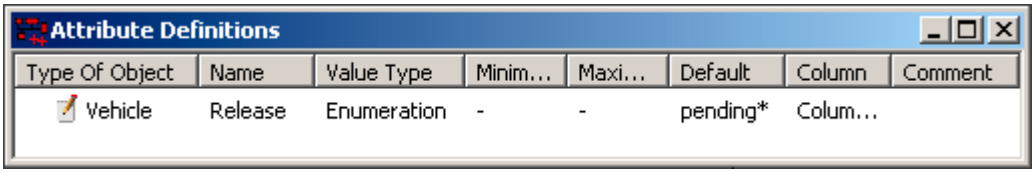

Figure 2-18: Attribute Definitions Window with an Attribute

Like system parameters, the values of user-defined attributes also appear in the columns on the right side of the Overview Window or in the corresponding List windows.

Note:	User-defined attributes of objects which still have their default values are identified by a * after the attribute value on the right side of the Overview Window.

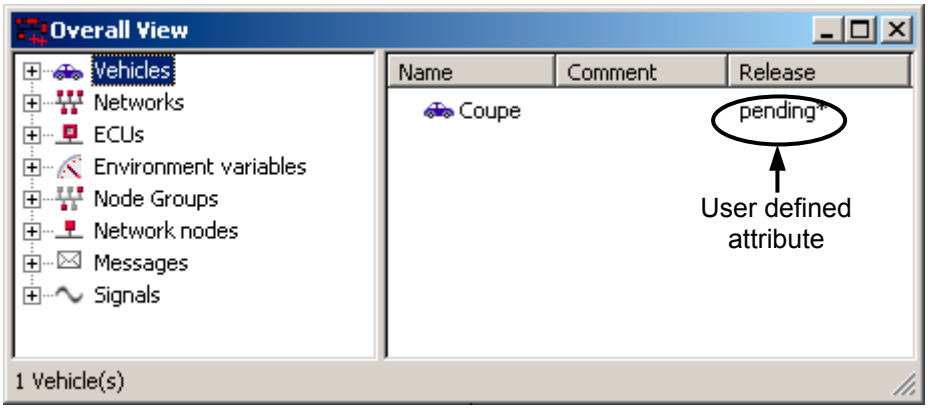

Figure 2-19: Overview Window with User-Defined Attribute

2.7.2 Modifying the Value of an Object's User-Defined Attribute

In the CAN database Tutorial.mdc or Tutorial.dbc set the user-defined attribute Release for the Vehicle object Coupe to the value 'denied'.

Like system parameters, the values of user-defined attributes can also be modified at any time.

2.7.2.1 Modifying the Values in the Object Dialog

Proceed as follows to modify the value of an object's user-defined attribute:

1. Select the object whose user-defined attribute value you wish to modify.
2. Choose the **Edit** command (**Edit** menu). The Object Dialog is opened.
3. Activate the **Attribute** page by clicking its tab.

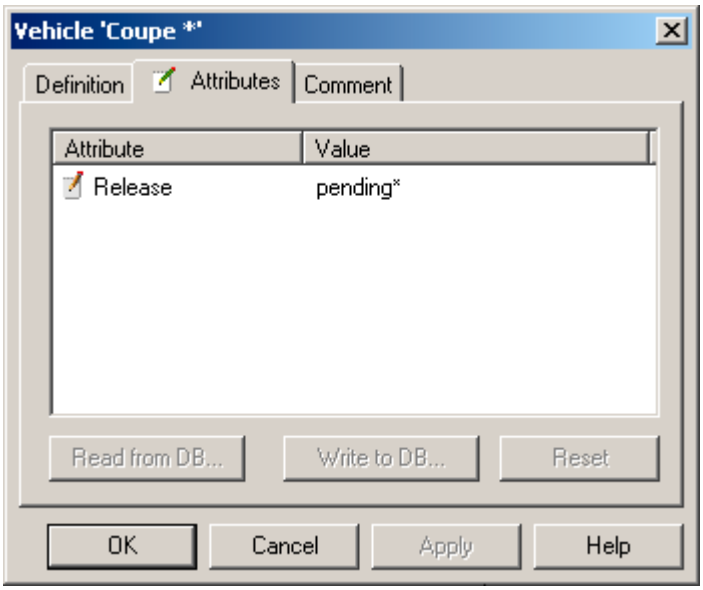

Figure 2-20: "Attributes" Page of the "Vehicle" Object Dialog

4. Activate the user-defined attribute whose value you wish to change by clicking the appropriate table cell. The corresponding table line is highlighted in color (Default: Blue).
5. Click the value of the attribute. A frame appears around the cell and if applicable the available attribute values are shown.
6. Enter the desired value or select it from the list.
7. Press the <Esc> key if you wish to abort the procedure, i.e. you wish to leave the value unchanged.
   Press the < ↵ Return> key if you wish to accept the changed value.
8. Press the **[OK]** button. The changed value of the object's user-defined attribute appears in the table on the right side of the Overview Window.

2.7.2.2 Modifying the Values in the Table in the Overview or List Window

Proceed as follows to modify the values of a user-defined attribute for an object directly in the table in the Overview Window or List Window:

1. Click the table cell containing the object's user-defined attribute to be modified. The corresponding line in the table is highlighted in color (Default: Blue). The activated cell is shaded with a different color (Default: Light gray).
2. Activate the editing mode for the activated cell by pressing the <F2> key or clicking the cell again. A frame appears around the cell and it is shaded in color (Default: Blue); if applicable a list of possible values is shown.
3. Change the attribute value by entering a different value or selecting a different option from the list shown.
4. Press the <Esc> key if you wish to abort the procedure, i.e. you wish to leave the parameter value unchanged.
   Press the < ↵ Return> key if you wish to accept the changed value.

2.8 Creating Variants of Existing Objects

In the CAN database Tutorial.mdc or Tutorial.dbc create a variant of the Vehicle object Coupe and assign the variant the name Sedan.

Proceed as follows to create a variant, i.e. a copy of an existing object, e.g. a Vehicle:

1. Select the object for which a variant should be created. This involves clicking the object in the Overview Window. The selected object is highlighted in color (Default: Blue).
2. Copy the selected object to the Clipboard with the **Copy** command (**Edit** menu).
3. Create the variant by pasting a copy of the object located in the Clipboard with the **Paste** command (**Edit** menu). The copy of the object appears in the Overview Window.
4. Modify the variant of the object (➔ Page 11).

Note:	After creating the variant of an object this variant is completely independent of the original object! Changes to the original object are not mirrored in the variant, and if desired these changes must be performed manually afterwards.

2.9 Consistency Check

Execute an automatic consistency check for the CAN database Tutorial.mdc or Tutorial.dbc with CANdb++ and correct any inconsistencies that are found.

To determine whether the objects of a CAN database and their interrelationships are consistent with one another, an automatic consistency check can be performed with CANdb++.

Proceed as follows to perform a consistency check:

- Choose the **Consistency check** command (**File** menu) to start the automatic consistency check. The results of the consistency check are displayed in a Consistency Check Window.
- If CANdb++ does not find any inconsistencies in the CAN database, then there will be no entries in the Consistency Check Window.
- However, if inconsistencies are found in the CAN database they will be listed in the Consistency Check Window.

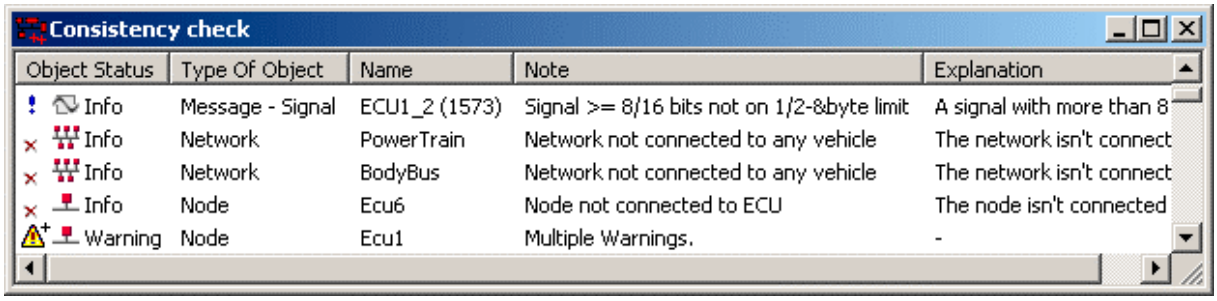

Figure 2-21: Consistency Check Window

Note:	When the Consistency Check Window is open it is updated automatically with each modification of the CAN database.

For Your Notes.

3 Version Administration

Note:	Version Administration for CAN databases and objects is only available in the CANdb++ Admin program version!

3.1 Preconditions for Version Administration Capability

To perform Version Administration the following preconditions must be satisfied:

- Microsoft® Visual Source Safe must be installed.
- The Version Administration archive in which the CAN database or object should be stored must have been created and must be accessible to the user.
- The configuration file SRCSAFE.INI must exist for the CAN database archive.
- The user must have read and write rights for all folders in which data are to be stored.

When these preconditions are satisfied CANdb++ Admin automatically establishes the connection to the Microsoft® Visual Source Safe archive, whose configuration file SRCSAFE.INI is entered in the **Archive Path** input box of the **Versioning** page (**Settings** dialog). Use the **Settings** command (**Options** menu) to open the **Settings** dialog.

3.2 Version Administration for CAN Databases and Objects

CANdb++ Admin permits direct access to CAN databases and objects that have been saved in Microsoft® Visual Source Safe.

The following actions can be executed directly, i.e. without leaving CANdb++ Admin:

- Opening a database that was saved in the Version Administration archive
- Reserving (Checking-out) a database
- Versioning (Checking-in) a database
- Displaying the version history of a database
- Comparing the different versions of a database
- Displaying the reservation status of a database
- Versioning (Checking-in) an object
- Displaying the version history of an object
- Comparing the different versions of an object

For Your Notes.

4 Appendix

4.1 File Name Extensions

The file name extension is understood to consist of the three characters located after the dot following the file name. The file name extension identifies the file type.

File Name Extension	File Type
CNT	Contents file for Help files (Contents)
DBC	CANdb network file (Data Base for CAN)
DLL	Run-time library (Dynamic Link Library)
EXE	Executable program (Executable Program)
HLP	Help file (Help)
INI	File with configuration options
MDC	CANdb++ CAN database
RPT	Crystal Report
TXT	ASCII text file
CSV	Text file with Comma/Character Separated Values

Table 4-3: File Name Extensions

4.2 INI Files

The INI files are located in the EXEC32 directory of your CANdb++ installation and contain configuration options for CANdb++. To make changes to options it is possible to edit the INI files with a ASCII editor.

Note:	You should exit CANdb++ before editing the INI files. The changed options in the INI files do not take effect in CANdb++ until CANdb++ is restarted.

4.2.1 VECTOR.INI

The following options can be set for CANdb++ in VECTOR.INI:

- Language of the menus and dialogs

Proceed as follows to change the language of the menus and dialogs:

1. Open the file VECTOR.INI with a ASCII editor.

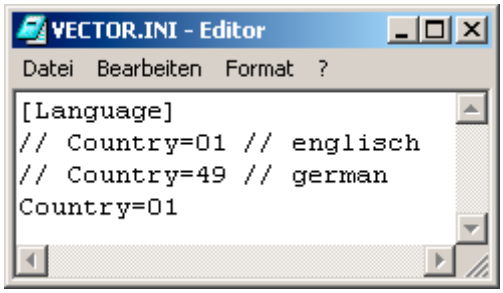

Figure 4-22: VECTOR.INI

2. In the `[Language]` section replace the language identifier entered in the `Country=` line by the identifier of the desired language.
   For example, English is configured in the following entry:
   ```
   [Language]
   Country=01
   ```
   Please refer to the comment lines (identified by //) of the `[Language]` section for the available languages.
3. Save the VECTOR.INI file and close the editor.

4.2.2 CANdb.INI

The following options can be configured for CANdb++ in CANdb.INI:

- Root directory for generated CAN drivers
  (section `[CAN_Driver]`, line "`RootDir=`")
- Display format for attributes of type Integer
  (section `[GUI_Settings]`, line "`DisplayIntAttrsAlwaysDecimal=`")
- Format to display signals with motorola byte ordering
  (section `[Format]`, line "`MotorolaFormat=`")
- Indexing of bits in layout dialog of messages
  (section `[Format]`, line "`UseInvertedLayoutIndexing=`")

Proceed as follows to make changes to these options:

1. Open the CANdb.INI file with a ASCII editor.

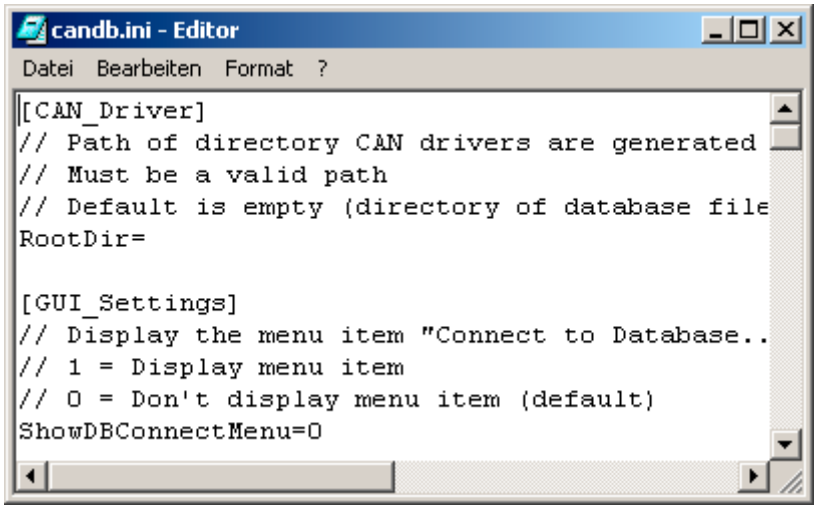

Figure 4-23: CANdb.INI

2. If you want to set the root directory for generated CAN drivers, enter the desired root directory in the line `RootDir=` of the section `[CAN_Driver]`.
   (When generating CAN drivers, sub-directories are created automatically in the root directory. The generated CAN drivers are stored in these sub-directories.)
3. Store the file CANdb.INI and close the editor.
   If CANdb++ was opened during editing, you have to exit CANdb++ and open it again to work with the new settings.

For Your Notes.

5 Index

P

S

T

U

V

W