

# OD Builder II

## Quick-start manual Version 1.00 (preliminary)

**Edition Mai 2016**

Document No.: L-1852e\_01

SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund  
Phone: +49 (3765) 38600-0 Fax: +49 (3765) 38600-4100  
Web: <http://www.systec-electronic.com> Mail: [info@systec-electronic.com](mailto:info@systec-electronic.com)



Product names used in this manual which are also registered trademarks have not been marked extra. The missing © mark does not imply that the trade name is unregistered. Furthermore it is not possible to determine the existence of any patents or protection of inventions on the basis of the names used.

The information in this manual has been checked carefully and is believed to be accurate. However, it is expressly stated that SYS TEC electronic GmbH does not assume warranty or legal responsibility or any liability for consequential damages which result from the use or contents of this user manual. The information contained in this manual can be changed without prior notice. Therefore, SYS TEC electronic GmbH will not accept any obligation.

Furthermore, it is expressly stated that SYS TEC electronic GmbH does not take warranty or legal responsibility or any liability for consequential damages which results from incorrect use of the hard- or software. The layout or design of the hardware can also be changed without prior notice. Therefore, SYS TEC electronic GmbH will not accept any obligation.

© Copyright 2016 SYS TEC electronic GmbH, D-08468 Heinsdorfergrund

All rights reserved. No part of this manual may be reproduced, processed, copied or distributed in any way without the express prior written permission of SYS TEC electronic GmbH.

Contact	Direct	Your Local Distributor
Address:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Please find a list of our distributors under:  <a href="http://www.systec-electronic.com/distributors">http://www.systec-electronic.com/distributors</a>
Ordering Information:	+49 (0) 37 65 / 38 600-0 <a href="mailto:info@systec-electronic.com">info@systec-electronic.com</a>	
Technical Support:	+49 (0) 37 65 / 38 600-0 <a href="mailto:support@systec-electronic.com">support@systec-electronic.com</a>	
Fax:	+49 (0) 37 65 / 38 600 4100	
Web Site:	<a href="http://www.systec-electronic.com">http://www.systec-electronic.com</a>	

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	Overview .....	7
<b>2</b>	<b>Installation and Program start.....</b>	<b>9</b>
2.1	Installation .....	9
2.2	Program start .....	15
<b>3</b>	<b>Use cases.....</b>	<b>16</b>
3.1	Create new project .....	16
3.1.1	Create new project from template .....	16
3.1.2	Create new project from EDS file .....	17
3.2	Load an existing project .....	19
3.3	Save a project .....	20
3.4	Edit a project .....	21
3.4.1	Modify a project item .....	21
3.4.2	Undo and Redo .....	21
3.4.3	Different panel views within the Editor Panel .....	21
3.4.3.1	Project Node Panel .....	22
3.4.3.2	Instance Node Panel.....	23
3.4.3.3	File Information Node Panel .....	24
3.4.3.4	Device Information Node Panel .....	25
3.4.3.5	Device Abilities Node Panel.....	27
3.4.3.6	Dummy Usage Node Panel .....	29
3.4.3.7	Comments Node Panel.....	30
3.4.3.8	Object List Node Panel .....	31
3.4.3.9	Object Node Panel.....	32
3.4.3.10	Sub-index Node Panel .....	36
3.4.4	Examples for creating objects manually.....	38
3.4.4.1	Creating a VAR object index as numeric type.....	38
3.4.4.2	Creating a VAR object index as string type.....	42
3.4.4.3	Creating a DOMAIN object index .....	43
3.4.4.4	Creating an ARRAY object index .....	44
3.4.4.5	Creating an object index as RECORD .....	49
3.4.4.6	Creating a new sub-index .....	50
3.4.5	Context Menu.....	51
3.4.6	Import templates .....	51
3.5	Verify a project .....	53
3.6	Export a project .....	53
3.7	Command line arguments .....	55
<b>4</b>	<b>Known issues .....</b>	<b>56</b>
	<b>Index .....</b>	<b>57</b>

## List of Tables

Table 1: Text box indicating correctness of input .....	21
Table 2: Controls of the project node panel .....	22
Table 3: Controls of instance node panel .....	23
Table 4: Controls of the file information node panel .....	25
Table 5: Controls of device information node panel .....	26
Table 6: List of controls of device information node panel linked with the OD .....	27
Table 7: Controls of the device abilities node panel .....	29
Table 8: Controls of dummy usage node panel .....	30
Table 9: Controls of comments node panel .....	31
Table 10: Controls of object list node panel .....	31
Table 11: Controls of object index node panel .....	34
Table 12: Differences of the access types .....	35
Table 13: Differences between the data locations .....	35
Table 14: Controls of sub-index node panel .....	38
Table 15: Available template files .....	52
Table 16: Export format types .....	54
Table 17: Command line options .....	55

## List of Figures

Figure 1: Setup – Welcome page .....	9
Figure 2: Setup – License Agreement page .....	9
Figure 3: Setup – Information page .....	10
Figure 4: Setup – User Information page .....	10
Figure 5: Setup – Select Destination Location page .....	11
Figure 6: Setup – Select Components page .....	11
Figure 7: Setup – Select Start Menu Folder page .....	12
Figure 8: Setup – Select Additional Tasks page .....	12
Figure 9: Setup – Ready to Install page .....	13
Figure 10: Setup – Installing page .....	13
Figure 11: Setup – Revision Information page .....	14
Figure 12: Setup – Completing page .....	14
Figure 13: Start screen of OD Builder II .....	15
Figure 14: About screen with detailed information .....	15
Figure 15: Menu command “File / New Project” .....	16
Figure 16: New project from template .....	16
Figure 17: Menu command “File / Save As” .....	17
Figure 18: Menu command “File / New Project from EDS” .....	17
Figure 19: EDS import summary dialog .....	17
Figure 20: Menu command “File / Open” .....	19
Figure 21: Project loaded screen .....	19
Figure 22: Menu command “File / Recently used” .....	20
Figure 23: Not saved project .....	20
Figure 24: Menu command “File / Save” .....	20
Figure 25: Saved project .....	20
Figure 26: Menu commands “Edit / Undo” and “Edit / Redo” .....	21
Figure 27: Project node panel .....	22
Figure 28: Instance node panel .....	23
Figure 29: File information node panel .....	24
Figure 30: Device information node panel .....	25
Figure 31: Apply settings to the OD .....	26
Figure 32: Device abilities node panel .....	27
Figure 33: Dummy usage node panel .....	29
Figure 34: Comments node panel .....	30
Figure 35: Object list node panel .....	31

Figure 36: Object index node panel.....	32
Figure 37: Sub-index node panel .....	36
Figure 38: Example VAR object index as numeric type.....	38
Figure 39: Example VAR object index as string type.....	43
Figure 40: Example DOMAIN object index .....	44
Figure 41: Example ARRAY object index .....	45
Figure 42: Example RECORD object index.....	49
Figure 43: Example sub-index.....	50
Figure 44: Context menu .....	51
Figure 45: Overwrite an existing object index.....	52
Figure 46: Import offset dialog.....	52
Figure 47: Output Window showing the verification result.....	53
Figure 48: Menu command Export .....	53
Figure 49: Save before export .....	54
Figure 50: Export dialog window .....	54
Figure 51: Output Window showing the export results .....	55

## List of abbreviations

CAN	Controller Area Network (according to ISO 11898-1:2003 and ISO 11898-2:2003)
CiA	CAN in Automation ( <a href="http://www.can-cia.org/">http://www.can-cia.org/</a> )
EDS	Electronic data sheet
GUI	graphic user interface
OD	Object Dictionary
OS	Operating System
PDO	Process data object
RAM	Random-Access Memory
ROM	Read-Only Memory
SDO	Service data object
tbd	to be defined

# 1 Introduction

Thank you that you have decided for the SYS TEC OD Builder II.

The OD Builder II is a Windows based configuration tool that was developed exclusively as support for the development of applications with the SYS TEC CANopen stack. This tool is running on Windows XP and newer Windows versions, and is characterized by its simple, user-friendly implementation based on a graphic user interface (GUI). It is based on the Microsoft .NET Framework 3.5.

The main application of the OD Builder II is the automatic generation of an Object Dictionary (OD) in the form of C source code files for a CANopen device, which then can be included seamlessly in an existing CANopen software project. Furthermore, it is possible to generate a so-called EDS file (electronic data sheet), which can be used in conjunction with various configuration tools for CANopen devices.

The functionality of this tool is two folded. First, this tool makes functions available for the development of a user specific Object Dictionary via step by step creation of the indexes and sub-indexes, whereby the individual index and sub-index entries should correspond to the CANopen standard CiA-301 V4.2, or the device profile assigned to the CANopen device in question.

Secondly, this tool offers an import interface for existing EDS files of a CANopen device. These files can be imported with the tool, whereby an Object Dictionary is generated based on the contents of the EDS file. Following this, an immediate generation of the Object Dictionary's corresponding C source code files for the CANopen project, or application specific adaptation of the Object Dictionary is possible.

Depending on the application, there are options for generating the C source code files as well as the related EDS file following configuration of the Object Dictionary. These files are prepared in a way that no additional user adaptations are necessary. The structure of the generated EDS file corresponds to the CANopen standard CiA-306 (Electronic Data Sheet Specification for CANopen, Version 1.3) and can be used as a basis for performing CANopen device conformance tests.

For more information, optional products, updates et cetera, we recommend to visit our website: <http://www.systec-electronic.com>. The content of this website is updated periodically and provides downloads of the latest releases of software and manuals.

## 1.1 Overview

The OD Builder II supports the following Features:

- Created for the .NET Framework 3.5. Therefore the tool can be used with all Windows OS versions where the .NET Runtime Libraries are installed.
- Creation of new projects based on a prepared template file.
- Modification of existing objects and sub-indices.
- Adding new objects and sub-indices.
- Copy and paste of single objects and sub-indices.
- Import of objects from prepared template files.
- Import of template files with an offset for realizing virtual devices in the device area of the OD.
- Selection of the objects and sub-indices that shall be exported via the export command.
- Verification of all settings within a project file, using a database file (based on CiA-301 V4.2.0.72).
- Import/Export of EDS files (according to CiA-306 V1.3).

- Export of \*.c/\*.h files for compilation with the SYS TEC CANopen stack.
- Creation of the VAR table used for indirect linking of application variables with the OD.



## 2 Installation and Program start

### 2.1 Installation

This section describes the installation of the OD Builder II.

The OD Builder II is delivered as an executable setup file for Microsoft Windows. The setup file is named: SO-1130.exe. To start the setup process, double click to this file. Firstly the *Welcome* page is shown as displayed in Figure 1.



Figure 1: Setup – Welcome page

Click to the *Next* button to continue. The *License Agreement* page is displayed (see Figure 2). Please read the license agreement carefully and click to “*I accept the agreement*”.

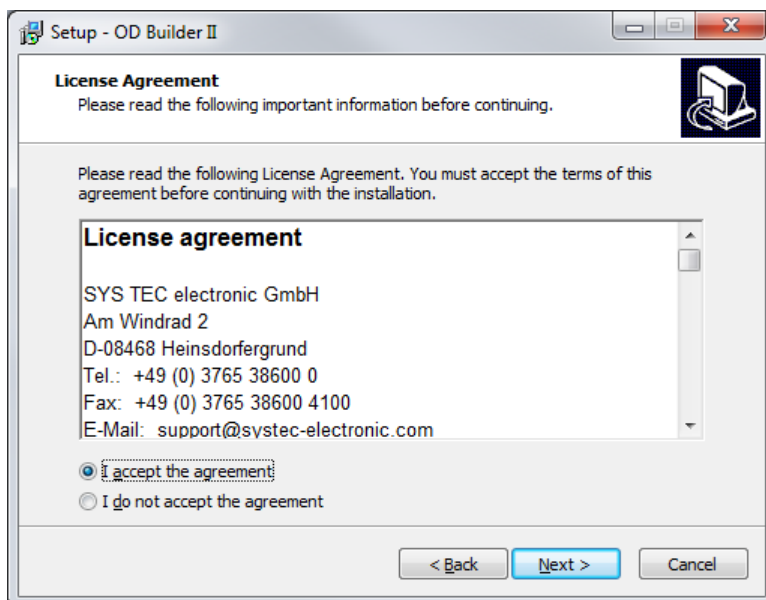


Figure 2: Setup – License Agreement page

After confirming the license agreement with the *Next* button, the *Information* page appears as displayed in Figure 3. This page notifies you about the version and release date of the OD Builder II. Additionally it informs you, that the Microsoft .NET Framework 3.5 is needed to be able to run the tool.

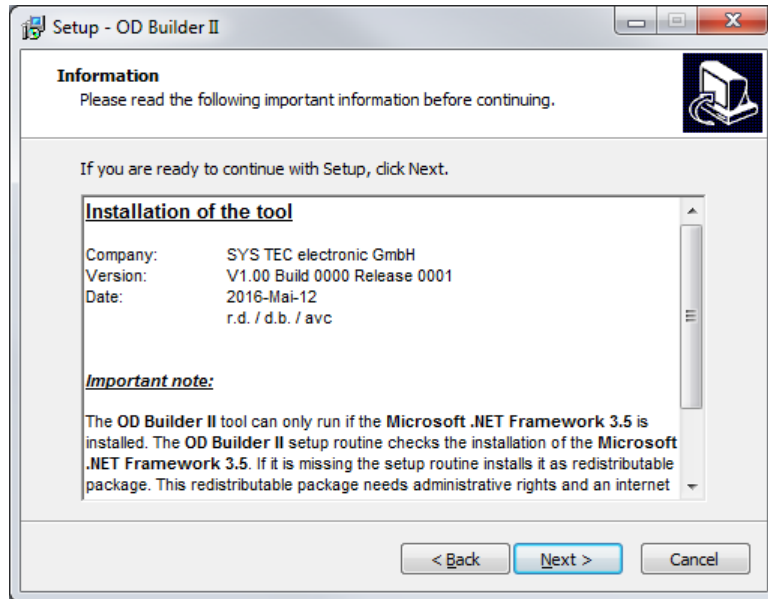


Figure 3: Setup – Information page

Click to the *Next* button. Now the *User Information* page is displayed (see Figure 4). Fill in your username and organisation. This information is only stored locally to your computer but not delivered via internet.

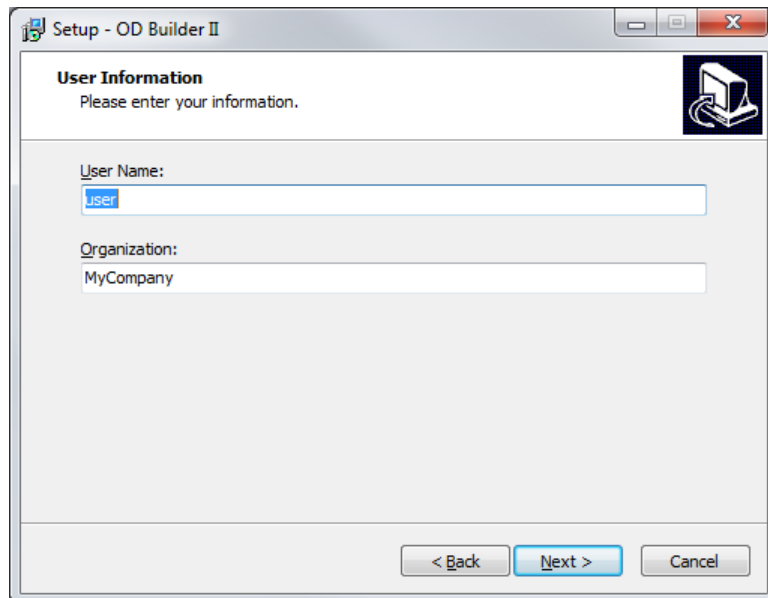


Figure 4: Setup – User Information page

Confirm your changes by clicking to the *Next* button. The *Select Destination Location* page is displayed (see Figure 5). You can type in an alternative destination location. You also can use the *Browse* button to select another destination location.

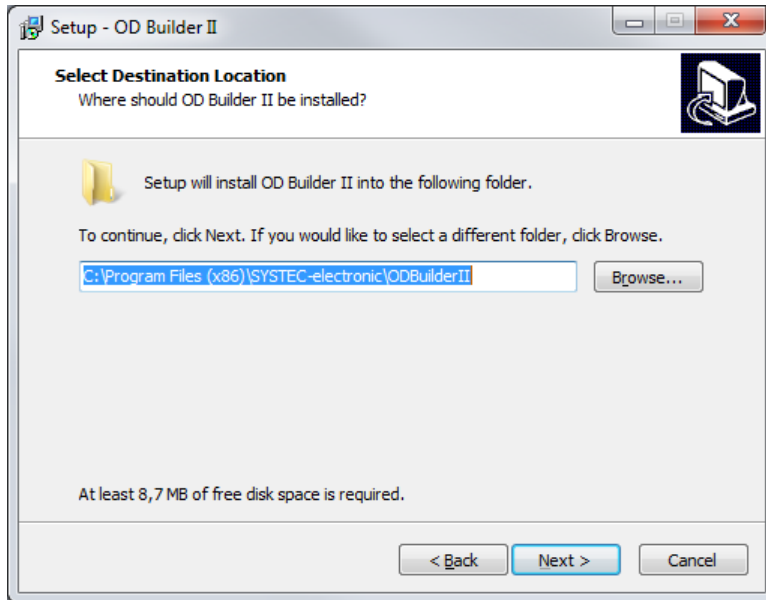


Figure 5: Setup – Select Destination Location page

After clicking to the *Next* button the *Select Components* page is displayed. Check or uncheck the components of your choice. The component *Program Files* cannot be unchecked.

**Note:**

The component *.NET Framework* is needed to run the OD Builder II tool. If it is already installed on your computer, the setup routine will not install it again. If it is not yet installed, then the Microsoft *.NET Framework 3.5 Redistributable Setup* is started. This setup needs administrative user rights and an internet connection. Possibly you need to restart the computer after the *.NET Framework 3.5 Setup* has finished.

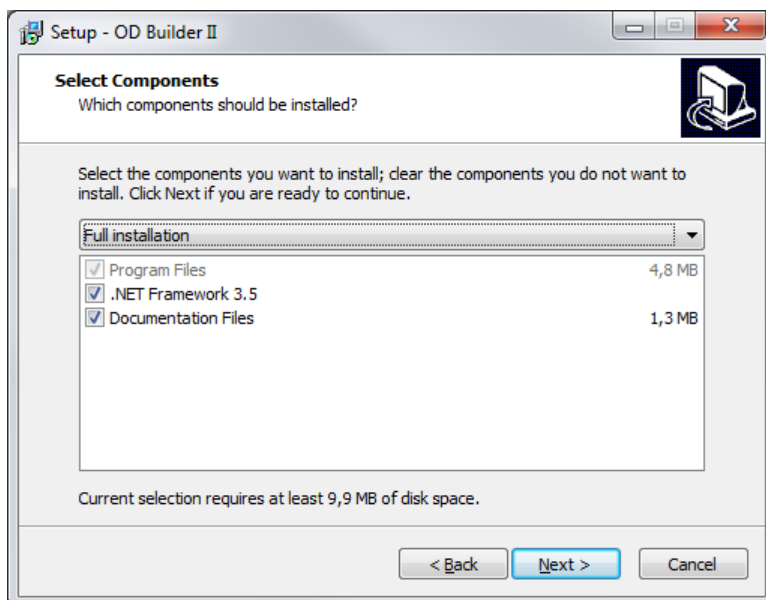


Figure 6: Setup – Select Components page

If all the needed components are selected, click to the *Next* button again. This displays the *Select Start Menu Folder* page. You may type in an alternative start menu folder than recommended. Click to the *Browse* button if you want to use an existing start menu folder.

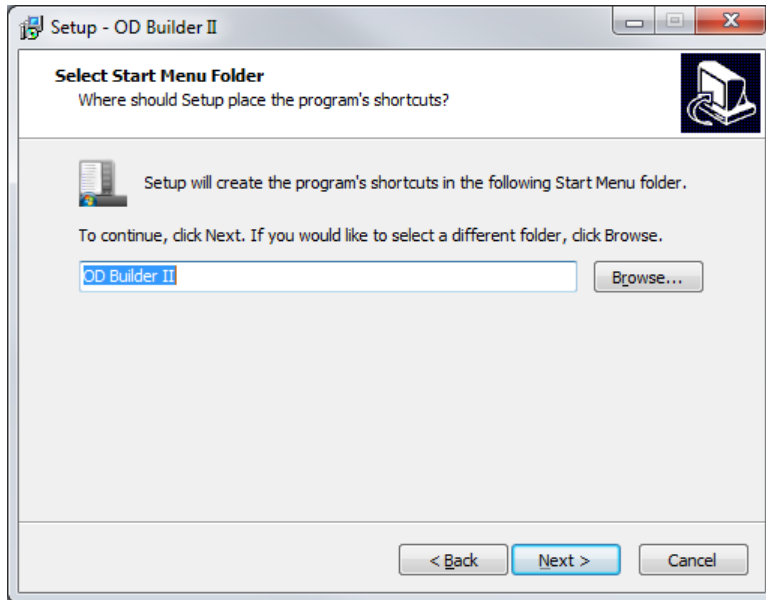


Figure 7: Setup – Select Start Menu Folder page

The click to the *Next* button confirms the selected start menu folder. The *Select Additional Tasks* page is displayed as shown in Figure 8. If the setup routine shall create a desktop icon for the OD Builder II tool, then please check the checkbox “*Create a desktop icon for OD Builder II*”.

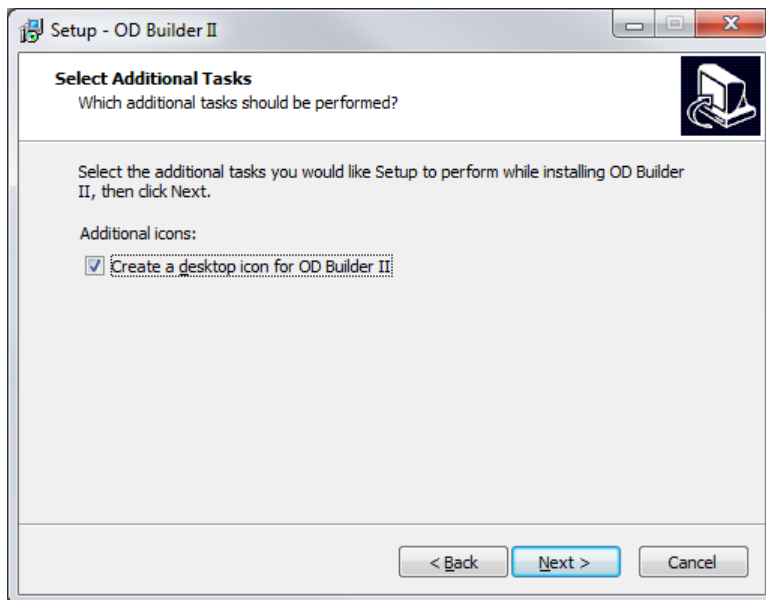


Figure 8: Setup – Select Additional Tasks page

The *Ready to Install* page is displayed after clicking the *Next* button (see Figure 9). This page summarizes all settings which you made for the setup routine. Click to the *Back* button if you decide to change any settings. Otherwise click to the *Next* button to confirm the settings and to start the setup process.

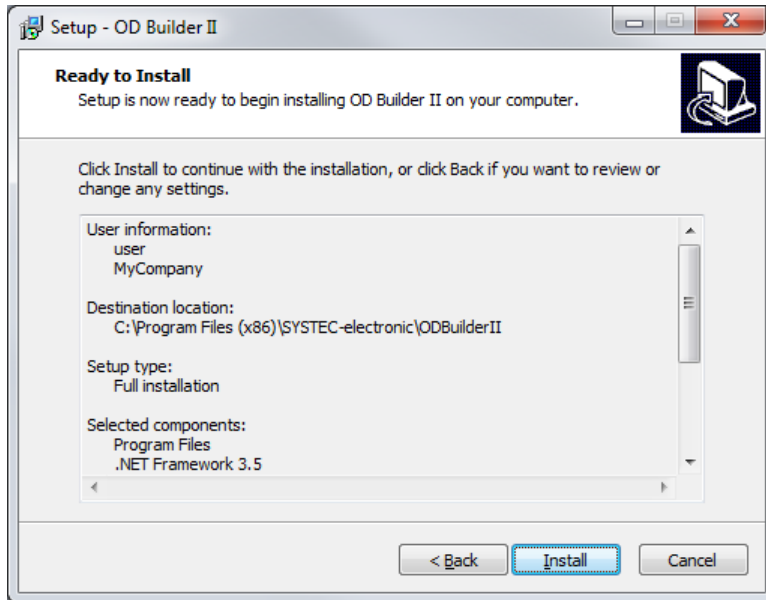


Figure 9: Setup – Ready to Install page

The *Installing* page is displayed in Figure 10. Here you may cancel the setup process by clicking to the *Cancel* button. Otherwise it finishes automatically and the *Revision Information* page is displayed as shown in Figure 11.

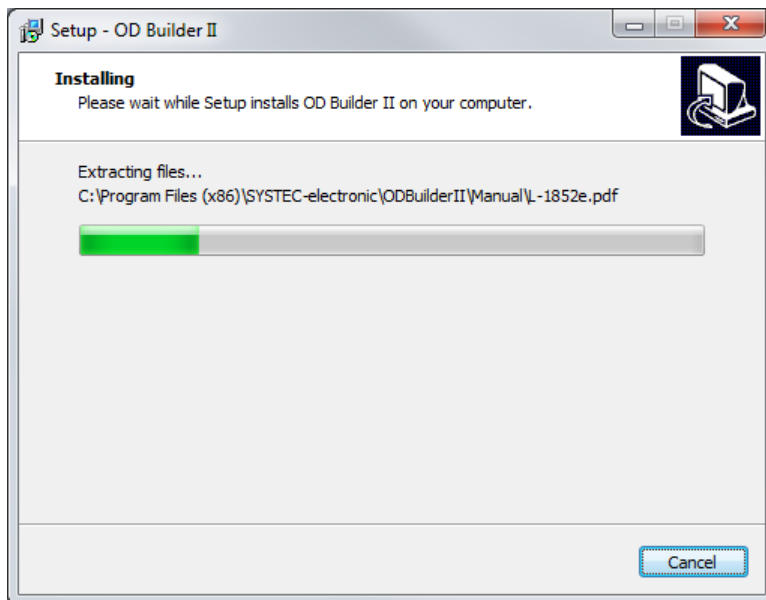


Figure 10: Setup – Installing page

The *Revision Information* page displays the latest changes of the OD Builder II. The content of the revision history is also stored to your hard disk. Open the file using the Windows Start Menu → All Programs → OD Builder II → Revision History.

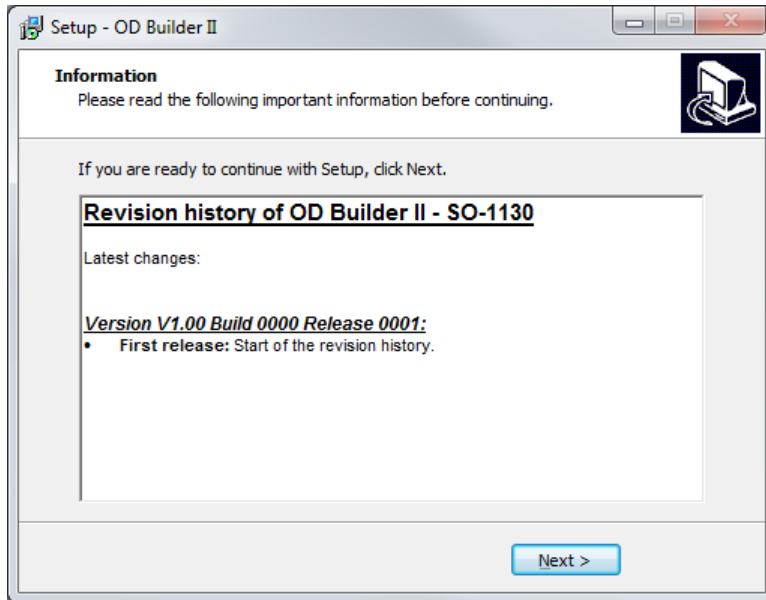


Figure 11: Setup – Revision Information page

By clicking the *Next* button the *Completing* page is displayed (see Figure 12). Close the setup routine by clicking to the *Finish* button.



Figure 12: Setup – Completing page

## 2.2 Program start

This section describes the program start of the OD Builder II. Figure 13 shows the start screen of OD Builder II. The object dictionary is displayed in a tree view on the left side (“OD tree”). The properties of a selected object dictionary element are shown in the top right area (“Editor panel”). Status messages, e.g., during export of an object dictionary to an EDS file, are shown in the output window in the bottom right area (“Output window”).

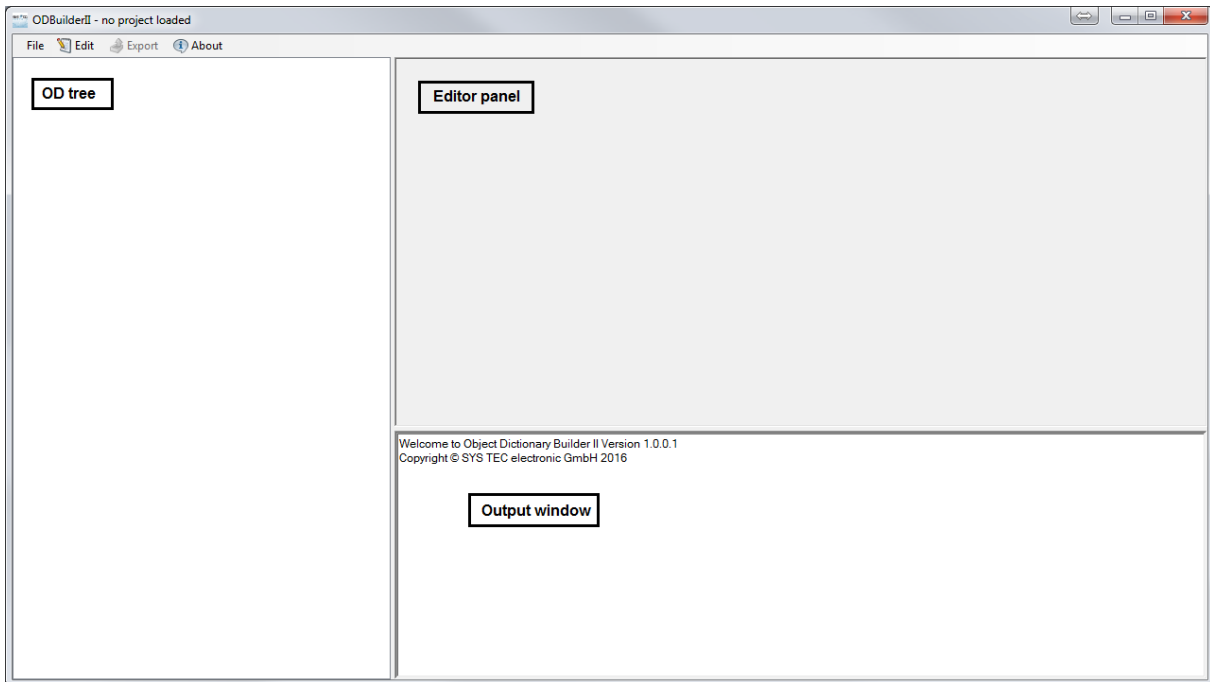


Figure 13: Start screen of OD Builder II

The output window displays the version number of the OD Builder II right after the start. More information about OD Builder II is shown by clicking “About” in the menu strip, which opens the dialog shown by Figure 14.



Figure 14: About screen with detailed information

### 3 Use cases

#### 3.1 Create new project

The OD Builder II offers the possibility to create a new object dictionary from scratch. You can start from a generic basic project or by importing an EDS file.

##### 3.1.1 Create new project from template

The common way to create a project from scratch is by clicking “File / New Project” in the menu strip (alternative: shortcut “ctrl + n”).

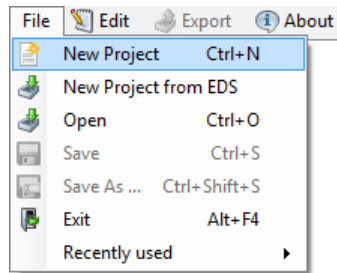


Figure 15: Menu command “File / New Project”

The OD Builder II loads a generic project named “NewProject”, see Figure 16. The object dictionary tree is fully expanded after creating a new project. Firstly type in a new project name for the new project.

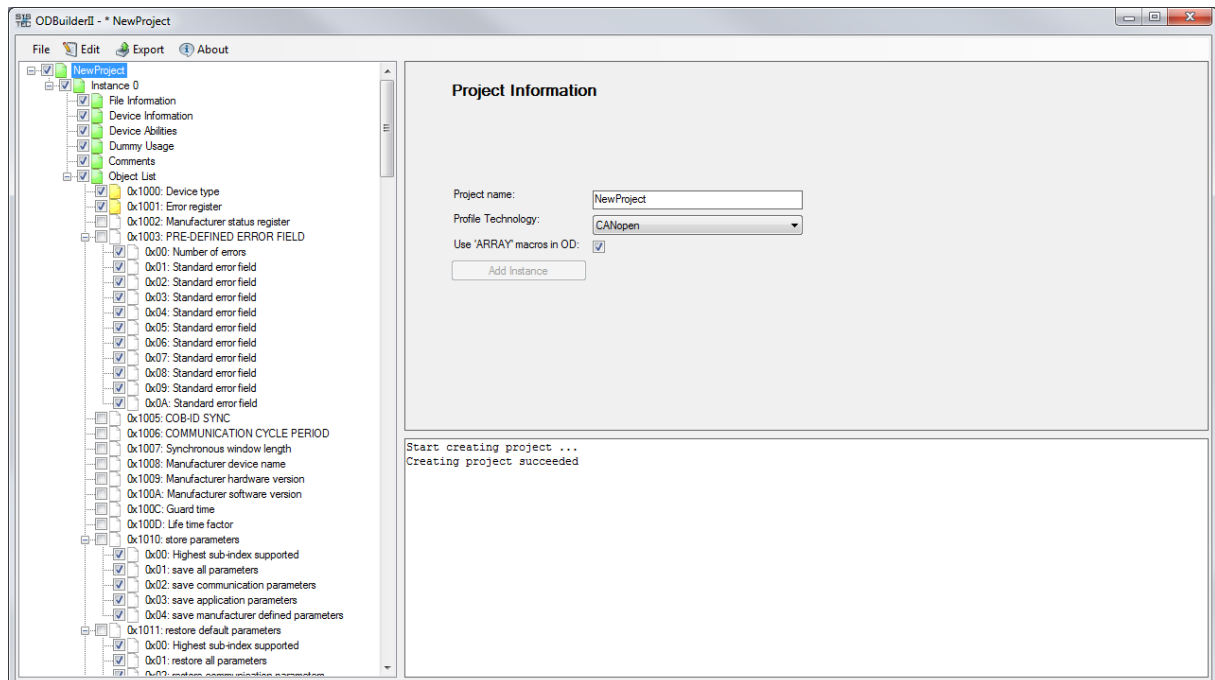


Figure 16: New project from template

The newly created project is not yet stored in a project file. To save the project, click “File / Save As ...” (alternative: shortcut “shift + ctrl + s”) to open a file save dialog.



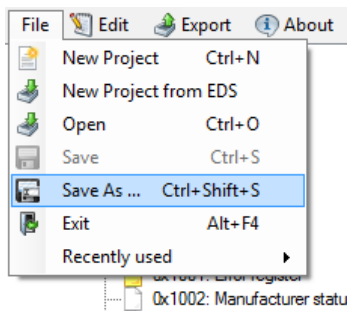


Figure 17: Menu command “File / Save As”

You have to provide a file name. OD Builder II project files use the file extension “.odbprj”. The full path to the currently loaded project file is shown in the title bar of the main window.

### 3.1.2 Create new project from EDS file

The second possibility to create a new project is using an EDS file as starting point. OD Builder II creates a new project based on the information and objects found in the EDS file. Click “File / New Project from EDS” in the menu strip.

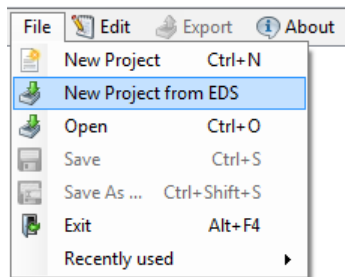


Figure 18: Menu command “File / New Project from EDS”

Choose the EDS file to import in the file open dialog. The imported project is verified. Error and warning information are shown in the output window. A dialog shows a summary and asks to continue, see Figure 19.

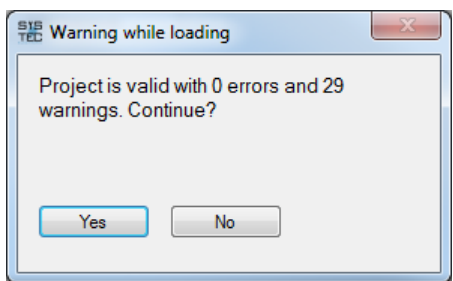


Figure 19: EDS import summary dialog

If you click “Yes”, the imported project is loaded but not saved. If you click “No”, the imported project is not loaded. The previous one keeps loaded.

**Note:**  
If there are errors, the erroneous objects are not imported, but all other objects are imported and displayed. An according error message is displayed in the output window. The user has to decide how to react to that:  
a) Ignore the message and the object is omitted.  
b) Add the object manually.



### 3.2 Load an existing project

Load an existing project by clicking “File / Open” (alternative: shortcut “ctrl + o”).

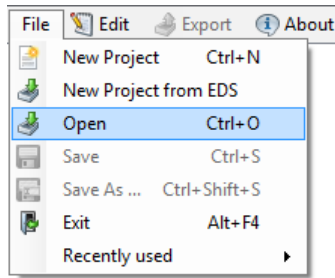


Figure 20: Menu command “File / Open”

The file dialog accepts files with file extension “.odbprj”. The project is verified during loading. Error and warning information are shown in the output window. Figure 21 shows an example. A project load summary dialog is also shown, compare to Figure 19. The object dictionary tree on the left side of the main window is completely expanded and the top most tree node is selected after loading an existing project.

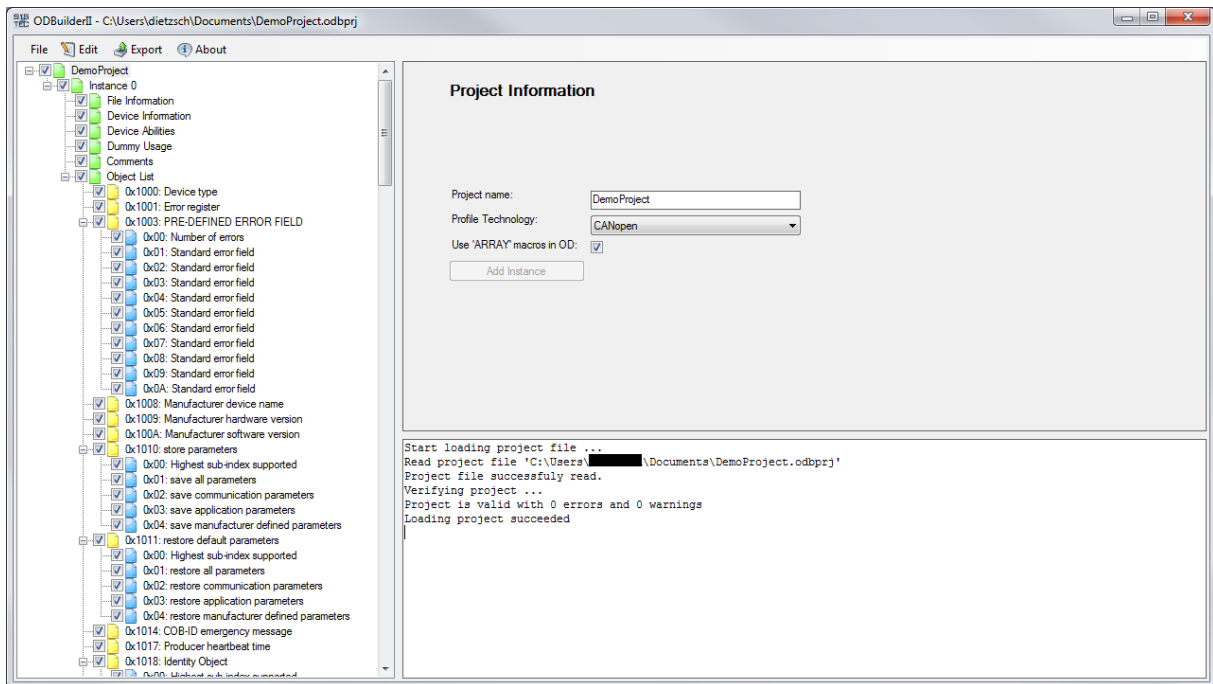


Figure 21: Project loaded screen

You can also choose a project file from “File / Recently used”, which provides the five recently used project files, see Figure 22. Project files that are not available anymore are coloured in grey and disabled.

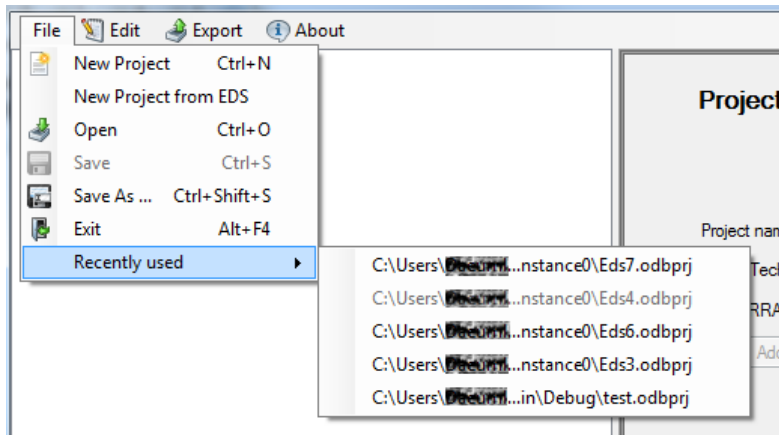


Figure 22: Menu command “File / Recently used”

### 3.3 Save a project

The title bar of the main window indicates if the currently loaded project is saved or not. If the project is modified but not saved, the project path has a preceding asterisk (“\*”-sign), see Figure 23.

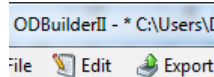


Figure 23: Not saved project

A project can be saved by selecting “File / Save” (shortcut: “ctrl + s”) or “File / Save As ...” shortcut “shift + ctrl + s”) in the menu strip. With the menu command “File / Save As ...” the project can be stored with a new file name.

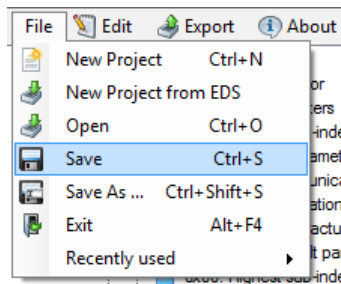


Figure 24: Menu command “File / Save”

If the project is saved, the asterisk is not present anymore in the title bar, see Figure 25.

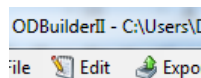


Figure 25: Saved project

### 3.4 Edit a project

#### 3.4.1 Modify a project item

A project item can be modified by selecting it in the object dictionary tree on the left side of the main window. The project item's properties are shown on the top right side of the main window. OD Builder II provides check boxes, drop down lists, date/time controls, numeric controls and text boxes to modify project item properties. Changing properties via check boxes, drop down lists, date/time controls and numeric controls are applied immediately. You cannot provide syntactically incorrect values by the controls mentioned above. The text box controls of OD Builder II provide a mechanism to indicate invalid inputs, see Table 1.

Description	Example
Text box input is correct and it is applied to project	File name: <input type="text" value="EdsFile"/>
Text box input is correct but not applied to project	File name: <input style="background-color: #90EE90;" type="text" value="EdsFile2"/>
Text box input is not correct but not applied to project	File name: <input style="background-color: #FFD700;" type="text" value="EdsFile eds"/>
Text box input is not correct but applied to project	File name: <input style="background-color: #FF0000;" type="text" value="EdsFile.ddd"/>

Table 1: Text box indicating correctness of input

**Note:**

Text box inputs are applied to the project by pressing the ENTER-key or if the text box loses the focus of the cursor. It is recommended to apply the text box input by the use of the ENTER-key.

#### 3.4.2 Undo and Redo

OD Builder II provides an Undo/Redo of up to ten operations. Click "Edit / Undo" (shortcut: "ctrl + z") or "Edit / Redo" (shortcut: "ctrl + r") in the menu strip of the main window to revert or repeat a modification of the project.

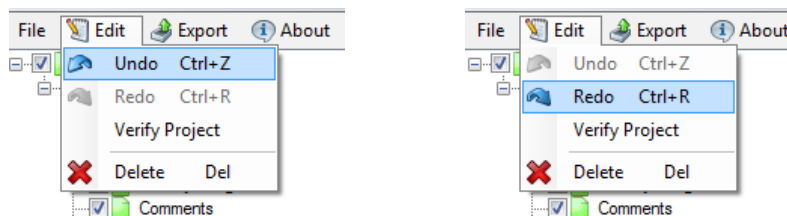


Figure 26: Menu commands "Edit / Undo" and "Edit / Redo"

**Note:**

The Redo command is only available if the Undo command was processed previously.

#### 3.4.3 Different panel views within the Editor Panel

Depending on the selected node within the project tree the right Editor Panel displays different views for applying changes to the node.

The following sub-sections describe the controls of the different panels.

### 3.4.3.1 Project Node Panel

If selecting the project node at the most top of the tree view the edit panel displays the project information as shown in Figure 27.

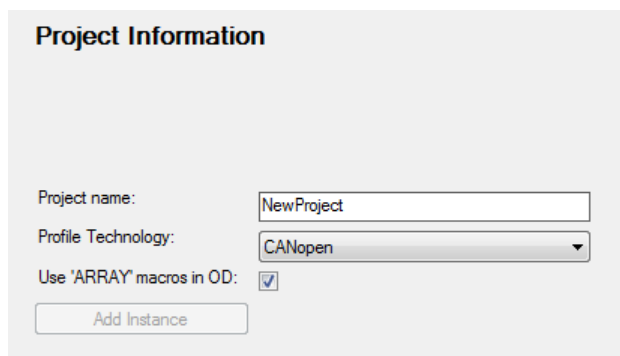


Figure 27: Project node panel

The project name is not part of any files to be exported. For that reason you can assign any name to the project. The controls are described in Table 2.

Label	Control type	Description
Project Name	Text box	General name of the project. This input is required and must not contain spaces.
Profile technology	Drop down list	Profile technology the project is built for. Currently only "CANopen" is supported.
Use ARRAY macro in OD	Check box	Configure the usage of the ARRAY-macro globally. If enabled, special macros are used for the ARRAY-object (global setting "OBD_IMPLEMENT_ARRAY_FCT" of CANopen Stack – refer to manual L-1020).
Add Instance	Button	Add a new instance to the project. Currently this is not supported.

Table 2: Controls of the project node panel

### 3.4.3.2 Instance Node Panel

Selection of the instance node displays instance information as shown in Figure 28.

The screenshot shows a panel titled "Instance Information" with the following fields and values:

- VAR table name: aVarTab\_g
- Node ID: 0x01
- OD file name: objdict
- OD init function name: ObdInitRam
- Instance number: 0

Figure 28: Instance node panel

All controls of the instance node panel are described in Table 3. Make sure that all settings are correct. Otherwise the object dictionary cannot be compiled with the SYS TEC CANopen stack. When creating a new project or importing an EDS file, all settings within the instance node panel are initialized with default values, so that the object dictionary files can be compiled with the included demos of the SYS TEC CANopen Stack.

Label	Control type	Description
VAR table name	Text box	Name of the VAR table of the object dictionary. This name is written to the file vartab.h while exporting the object dictionary as *.c/*.h file format. It must not contain spaces. The default name used by the demos in SYS TEC CANopen Stack (SO-877) is "aVarTab_g".
Node ID	Text box	Node ID of the CANopen instance. This input must be a valid decimal or hexadecimal number in the range from 0x01 to 0x7F.
OD file name	Text box	Name of the c-file of the object dictionary without file extension. This input must not contain spaces. The default name used by the demos in the CANopen Stack is "objdict". For CANopen applications with multiple instances the OD file names must have different names for each instance.
OD init function name	Text box	Name of the initialization function of the object dictionary. This input must not contain spaces. The default name used by the demos in the CANopen Stack is "ObdInitRam". For CANopen applications with multiple instances the OD initialization function names must have different names for each instance.
Instance number	Label	Index number of the object dictionary instance. This label cannot be changed.

Table 3: Controls of instance node panel

### 3.4.3.3 File Information Node Panel

If the node “File information” is selected, the right Editor Panel displays the file information node panel as shown in Figure 29.

**EDS File Information**

File name:	<input type="text" value="EdsFile"/>	Created by:	<input type="text" value="User Name"/>
File version:	<input type="text" value="1.0"/>	Creation date:	<input type="text" value="2016-05-10"/>
EDS version:	<input type="text" value="4.2"/>	Creation time:	<input checked="" type="checkbox"/> 08:57:43
Format name:	<input type="text" value="CANopen"/>	Modified by:	<input type="text"/>
		Modification date:	<input type="text" value="2016-05-10"/>
		Modification time:	<input type="text" value="09:38:43"/>

Figure 29: File information node panel

All information (except the “Format name”) is written to the section “FileInfo” of the EDS file. When creating a new project any information is filled with default values. Only the values “Modified by”, “Modification date” and “Modification time” keeps empty. You can apply your own changes here. For detailed information for each control refer to Table 4



Label	Control type	Description
File name	Text box	Name of the EDS file
File version	Text box	File version of the EDS file in format "x.y" (x,y = (0..255)). This version is written to the EDS file section [FileInfo]. Value x is written to the keyword "FileVersion" (UNSIGNED8) and value y is written to the keyword "FileRevision" (UNSIGNED8).
EDS version	Combo box	Indicates the version of the specification. This combo box is read-only – the current version only supports the version "4.2".
Format name	Text box	Specifies the profile technology of the EDS file. This text box is read-only because the profile technology is specified within the project node panel (refer to section 3.4.3.1).
Created by	Text box	Author of the EDS file (max. 245 characters).
Creation date	Date time picker	Specifies the creation time in format "hh:mm:ss" (24 hours).
Creation time	Date time picker	Specifies the creation date in format "yyyy-MM-dd".
Modified by	Text box	Author of the last modifications of the EDS file (max. 244 characters).
Modification date	Date time picker	Specifies the last modification date in format "yyyy-MM-dd". This entry is optional for EDS files. Set the check box within this control if this entry shall be written to the EDS file via the export command.
Modification time	Date time picker	Specifies the last modification time in format "hh:mm:ss" (24 hours). This entry is optional for EDS files. Set the check box within this control if this entry shall be written to the EDS file via the export command.

Table 4: Controls of the file information node panel

### 3.4.3.4 Device Information Node Panel

If the node "Device information" is selected within the project tree, the right Editor Panel displays the device information node panel as shown in Figure 30.

The screenshot shows a form titled "Device Information" with the following fields:

- Vendor name: DeviceVendor
- Vendor ID: 0x00000000
- Device family: DeviceFamily
- Product name: ProductName
- Product ID: (empty)
- Order number: 0
- Software version: V1.00
- Firmware version: V1.00
- Hardware version: V1.00
- Build date: 2016-05-10 (with a calendar icon)
- Specification revision: 1.0

Figure 30: Device information node panel

All the information of the device information node panel is written to the EDS file's section "DeviceInfo" except the values for "Software version", "Firmware version", "Hardware version", "Build date" and

“Specification revision”. All the omitted values are prepared for exporting of XDD files which will be supported in further versions of the OD Builder II tool. The controls of the device information node panel are described in Table 5.

Label	Control type	Description
Vendor name	Text box	Specifies the vendor name (max. 244 characters).
Vendor ID	Text box	Specifies the unique vendor ID according to the identity object 0x1018 sub-index 0x01 (UNSIGNED32). This value can be specified in decimal format as well as in hexadecimal format.
Device family	Text box	States the family of the device (not needed for EDS files).
Product name	Text box	Specifies the product name (max. 243 characters).
Product ID	Text box	Specifies the manufacturer product code according to the identity object 0x1018 sub-index 0x02 (UNSIGNED32). This value can be specified in decimal format as well as in hexadecimal format.
Order number	Text box	Specifies the manufacturer order code for this product (max. 245 characters).
Software version	Text box	Specifies the software version for this product according to the object 0x100A. There are no format limits for the version. This value is not written to the EDS file.
Firmware version	Text box	Specifies the firmware version for this product. There are no format limits specified for the firmware version. This value is not written to the EDS file.
Hardware version	Text box	Specifies the hardware version for this product according to the object 0x1009. There are no format limits specified for the hardware version. This value is not written to the EDS file.
Build date	Date time picker	Specifies the build date of the software unit. This entry is not written to EDS files.
Specification revision	Text box	Specifies the revision of the specification, the device conforms to. This value is not written to EDS files.

Table 5: Controls of device information node panel

Some controls of the device information node panel are linked with the according default values objects in the OD. If they are changed, the OD Builder II asks whether the appropriate object shall be changed too (see Figure 31). Table 6 lists all controls and the according linked objects within the OD.

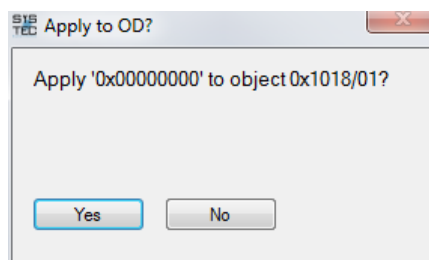


Figure 31: Apply settings to the OD

Control label	Linked object index	Linked sub-index
Vendor ID	0x1018	1
Product ID	0x1018	2
Software version	0x100A	(0)
Firmware version	(0x100A)	(0)
Hardware version	0x1009	(0)

Table 6: List of controls of device information node panel linked with the OD

### 3.4.3.5 Device Abilities Node Panel

#### Device Abilities

Supported Baud Rates:

10 kBit/s     250 kBit/s

20 kBit/s     500 kBit/s

50 kBit/s     800 kBit/s

100 kBit/s     1000 kBit/s

125 kBit/s     Auto

General Features:

SDO client

Support MPDO

Self Start Device

Boot-up slave

LSS Slave Support

Num Rx PDO:

Num Tx PDO:

PDO Granularity:

Master Features:

SDO manager

Configuration manager

Flying master

Boot-up master

LSS Master Support

Device Commissioning:

CANopen Manager

Node ID:

Node name:

Act. Baud Rate:

Network num:

Network name:

Figure 32: Device abilities node panel

Label	Control type	Description
10 kBit/s – 1000 kBit/s Auto	Check boxes	Indicates the supported baud rates.
SDO client	Check box	Indicates whether the CANopen device supports the SDO client (at least one object exists and is exported within the range of 0x1280 to 0x127F according to the CiA-301 standard). This check box is read-only and automatically updated. The value is not written to the EDS file.
Support MPDO	Check box	Indicates whether the CANopen device supports multiplexed PDOs.
Self Start Device	Check box	States the support of self-starting device functionality. A self-starting device is a device which enters the NMT state OPERATIONAL by itself without the need of a CANopen master sending the NMT command "Enter Operational". This value is not written to EDS files.
Boot-up slave	Check box	Indicates the simple boot-up slave functionality.
LSS Slave Support	Check box	Indicates if LSS slave functionality is supported.
Num Rx PDO	Text box	Indicates the number of supported receive PDOs. Here at least one object exists and is exported within the range of 0x1400 to 0x15FF (respectively 0x1600 to 0x17FF) according to the CiA-301 standard. This text box is read-only and updated automatically.
Num Tx PDO	Text box	Indicates the number of supported transmit PDOs. Here at least one object exists and is exported within the range of 0x1800 to 0x19FF (respectively 0x1A00 to 0x1BFF) according to the CiA-301 standard. This text box is read-only and updated automatically.
PDO Granularity	Combo box	Specifies the minimum granularity in bits allowed for the PDO mapping on this CANopen device. This combo box is read-only. The current version of OD Builder II only supports the granularity of 8 bits.
SDO manager	Check box	Indicates whether the CANopen device is able to act as an SDO manager (at least one object exists and is exported within the range of 0x1F00 to 0x1F11 according to the CiA-302-5 standard). This value is not written to the EDS file.
Configuration manager	Check box	Indicates whether the CANopen device is able to act as a configuration manager (at least one object exists and is exported within the range of 0x1F20 to 0x1F27 according to the CiA-302-3 standard). This value is not written to the EDS file.
Flying master	Check box	States the support of flying master functionality (the object 0x1F90 exists and is exported according to the CiA-302-2 standard). The value is not written to the EDS file.
Boot-up master	Check box	States the support of boot-up master functionality. If the CANopen device conforms to the CiA-301 standard the value is written to the EDS file as keyword "SimpleBootUpMaster" in section "DeviceInfo".
LSS Master Support	Check box	States the support of layer setting services as a master.

Label	Control type	Description
CANopen Manager	Check box	Indicates whether the CANopen device is able to act as a CANopen manager. The value is not written to the EDS file.
Node ID	Text box	Specifies the node ID of the CANopen device. The value is the very same as specified within the instance node panel. This text box is read-only and updated automatically if the node ID is changed within the instance node panel.
Node name	Text box	Specifies the name of the CANopen device. The value is combined with the value specified in object index 0x1008 (Manufacturer device name). If the "Node name" is changed, the OD Builder II asks whether index 0x1008 should be changed too.
Act. Baud rate	Combo box	Value of the actual baud rate. Note: This Combo box is filled dynamically with the baud rates that are supported (see check boxes "10 kBit/s" to "1000 kBit/s". If none of these check boxes is selected, this combo box can't be changed. The value is not written to the EDS file.
Network num	Text box	The unique number of the network segment, the CANopen device is connected to. The value is not written to the EDS file.
Network name	Text box	The name of the network segment, the CANopen device is connected to. The value is not written to the EDS file.

Table 7: Controls of the device abilities node panel

### 3.4.3.6 Dummy Usage Node Panel

Selecting the Dummy Usage node the Editor Panel shows the checkboxes for the dummy entries, see Figure 33. Dummy entries are dummy object indexes (representing data types) which can be mapped into receive PDOs if the CANopen device does not need the data value at this position of the PDO data.

Check one or more checkboxes to specify which of the data types can be used as dummy entry within a receive PDO in your device.

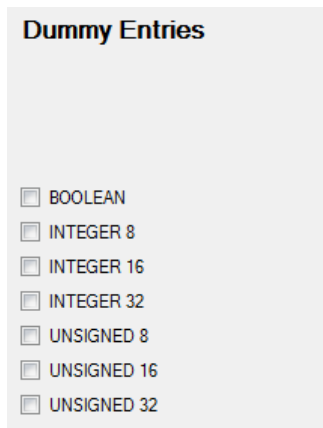


Figure 33: Dummy usage node panel

Label	Control type	Description
BOOLEAN	Check box	Indicates whether the data type BOOLEAN (index 0x0001) can be mapped to receive PDOs as dummy entry.
INTEGER 8	Check box	Indicates whether the data type INTEGER8 (index 0x0002) can be mapped to receive PDOs as dummy entry.
INTEGER 16	Check box	Indicates whether the data type INTEGER16 (index 0x0003) can be mapped to receive PDOs as dummy entry.
INTEGER 32	Check box	Indicates whether the data type INTEGER32 (index 0x0004) can be mapped to receive PDOs as dummy entry.
UNSIGNED 8	Check box	Indicates whether the data type UNSIGNED8 (index 0x0005) can be mapped to receive PDOs as dummy entry.
UNSIGNED 16	Check box	Indicates whether the data type UNSIGNED16 (index 0x0006) can be mapped to receive PDOs as dummy entry.
UNSIGNED 32	Check box	Indicates whether the data type UNSIGNED32 (index 0x0007) can be mapped to receive PDOs as dummy entry.

Table 8: Controls of dummy usage node panel

### 3.4.3.7 Comments Node Panel

An EDS file can include comments. Using the comments node panel these comments can be edited or cleared. Figure 34 shows the comments node panel, Table 9 describes its controls.

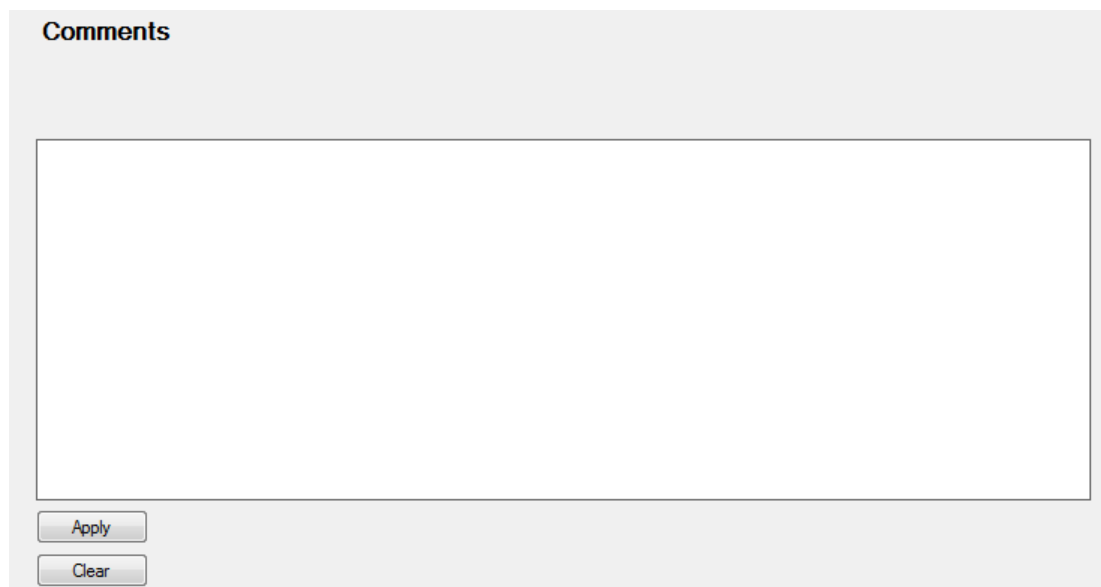


Figure 34: Comments node panel

Label	Control type	Description
Comments	Text box	Specifies the comments written to the EDS file section “Comments”. Each line specifies a new line keyword within the comments section. The maximum length of a line is 249 characters. The maximum number of lines is limited by the maximum number of characters of this edit box: 32767 (including CRLF).
Apply	Button	Click the “Apply” button to apply all changes of the comments text box.
Clear	Button	To clear the whole content of the comments text box click the button “Clear”.

Table 9: Controls of comments node panel

### 3.4.3.8 Object List Node Panel

The Object List node lists all object indices of the CANopen instance as child nodes. Objects marked with a yellow symbol  are exported with the “Export” command, objects tagged with a white symbol  are not exported.

If the Object List node is selected, the right Editor Panel has the content as shown in Figure 35. Its controls are described in Table 10.

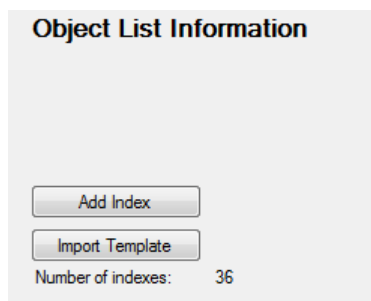


Figure 35: Object list node panel

Label	Control type	Description
Add Index	Button	Click to add a new object index (see section 3.4.4.1 and the following).
Import Template	Button	Click to import a whole template of object indexes (see section 3.4.6).
Number of indexes	Label	Shows the number of object indexes of the whole CANopen instance (objects which are exported as well as not exported).

Table 10: Controls of object list node panel

### 3.4.3.9 Object Node Panel

Export object

Object index:  Data type:  Default Value:

Object name:  Access type:   Use Value Range

Callback handler:  Data location:  Lower Limit:

Object type:  Number of subindices:  Upper Limit:

Category:

Index location in RAM  Use ARRAY macro for object index

Write default value to EDS file  Sub index 0x00 is read-write

Write default value to objdict.h

PDO mappable  Store to non-volatile memory

Treat "ro" as "const"

Variable name:

Reserved string size:

Figure 36: Object index node panel

Note that the state of some controls of the object index panel is changed dynamically. For example controls can be hidden or coloured grey (read-only). This state depends on the settings done before (e.g. object type, data type, access type, ...).



Label	Control type	Description
Export	Check box	If the “Export” check box is checked, the object index is exported via the “Export” command. Otherwise this object only exists as place holder (comparable with an out-commented function call in a C/C++ file). This check box is linked to the check box within the OD tree view at the selected object index node.
Object index	Text box	Specifies the object’s index value within the range 0x1000 to 0xFFFE.
Object name	Text box	Specifies the name of the object index.
Callback handler	Text box	The callback handler which is called by the CANopen stack on any access event (SDO or reading/writing by the CANopen application via CcmReadObject()/CcmWriteObject() – refer to manual L-1020). If this text box is read-only (coloured grey) the callback handler is defined within the CANopen stack. Otherwise the CANopen application has to define it.
Object type	Combo box	Specifies the object type of the object index. Possible values are: VAR, DEFTYPE, RECORD, DEFSTRUCT, ARRAY and DOMAIN. The object types VAR and DEFTYPE define an object without sub-indices. With the object types RECORD, DEFSTRUCT and ARRAY the object index contains sub-indices. A DOMAIN defines an object with a huge data size (sometimes a variable size of data).
Category	Combo box	The category specifies the location of the object within an EDS file. Possible values are: <i>mandatory</i> , <i>optional</i> and <i>manufacturer</i> . A mandatory object is located within the section <i>MandatoryObjects</i> of the EDS file (objects 0x1000, 0x1001 and 0x1018). Optional objects are located within the section <i>OptionalObjects</i> and manufacturer objects are located within the section <i>ManufacturerObjects</i> (objects within the range 0x2000 to 0x5FFF).
Index location in RAM	Check box	Specifies whether the object’s definition shall be located in RAM instead of ROM. This is only necessary if the object’s definition (e.g. data type) can be changed in run-time.
Write default value to EDS file	Check box	Specifies whether the default value shall be written to the EDS file.
Write default value to objdict.h	Check box	Specifies whether the default value shall be written to the objdict.h file to be compiled with the CANopen stack. If this check box is not checked the OD macro in objdict.h is used with the suffix “_NOINIT”. In this case the CANopen stack does not write any value on NMT event <i>Reset Node</i> or <i>Reset Communication</i> . Do not uncheck this check box for objects with access type = CONST!
PDO mappable	Check box	Specified whether the object can be mapped into a PDO. This control is only available for the object types VAR, DEFTYPE and for numeric data types.
Data type	Combo box	Specifies the data type of the object.
Access type	Combo box	Specifies the access type of the object as written to the EDS file. Possible values are: <i>constant</i> , <i>read-only</i> , <i>read-write</i> , <i>read-write process input</i> , <i>read-write process output</i> , <i>write-only</i> . (refer to Table 12).
Data location	Combo box	Specifies the data location of the object’s actual data value. Refer to Table 13 for all possible values.

Label	Control type	Description
Number of subindices	Numeric up/down	For ARRAY objects this control specifies the number of sub-indices (including sub-index 0x00). The value needs to be within the range of 2 to 255.
Use ARRAY macro for object index	Check box	Specifies whether the special ARRAY macro shall be used for this object when exporting to the file objdict.h (one macro for all sub-indices). If unchecked each sub-index is written to the file objdict.h. The special ARRAY macro is only used if the global setting is checked within project node panel.
Sub index 0x00 is read-write	Check box	Especially the object index 0x1003 is defined as an ARRAY in CiA-301. Usually the sub-index 0 of an ARRAY is defined with the access type "const". But sub-index 0 of object 0x1003 has to be "read-write" according to CiA-301. Check this check box for all ARRAY objects whose sub-index 0 shall be "read-write" instead of "const".
Add sub-index	Button	This button is only available for entries of object type DEFSTRUCT or RECORD (see section 3.4.4.5). Click this button to add a new sub-index.
Store to non-volatile memory	Check box	Specifies whether the object's actual data value shall be stored to the non-volatile memory if the "save" command is written to object 0x1010 (Store/Restore).
Treat "ro" as "const"	Check box	The object's access type is specified as to be written to the EDS file. All access types written to the EDS file have to be defined as defined within the CiA standards. But for the SYS TEC CANopen stack a "read-only" object can be treated as "const" object to reduce the RAM usage.
Default value	Text box	Specifies the default value for a DEFTYPE or VAR object. The default value is set to the object's actual data value after initialization and after the NMT command <i>Reset Communication</i> and/or <i>Reset Node</i> .
Use Value Range	Check box	If this check box is checked the lower and upper limit can be defined for the object's data value.
Lower Limit	Text box	Specifies the lower limit for the object's data value.
Upper Limit	Text box	Specifies the upper limit for the object's data value.
Variable name	Text box	For objects with data location "application direct" or "application indirect" (see Table 13) this text box specifies the variable provided by the application, the object is linked to. All OD accesses to the object's actual data value are redirected to the specified variable name. The variable has to be specified in C notation. For ARRAY objects no brackets are allowed.
Reserved string size	Numeric up/down	In case the object's data type is a string and the access type includes the write access rights then this value specifies the maximum length in characters (including the \0-termination) for the string held in RAM.

Table 11: Controls of object index node panel

Access type	memory location (actual Data)	SDO write access	SDO read access	app write access	app read access	PDO mappable <sup>1</sup>
constant	ROM	no	yes	no	yes	no
read-only	RAM	no	yes	yes	yes	yes
read-write	RAM	yes	yes	yes	yes	yes
read-write process input	RAM	yes	yes	yes	yes	yes <sup>2</sup>
read-write process output	RAM	yes	yes	yes	yes	yes <sup>2</sup>
write-only	RAM	yes	no	yes	no	yes

Table 12: Differences of the access types



<sup>1</sup> Ability of PDO mapping according to the SYS TEC CANopen stack SO-877/SO-1063

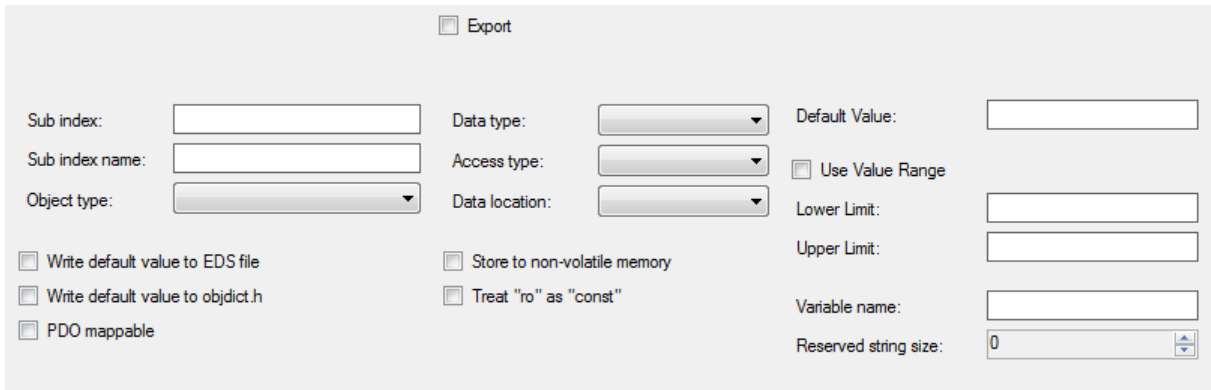
<sup>2</sup> According to EDS file standard the access type needs to be “read-write process input/output” for being PDO mappable. Only use these access types for objects which can be mapped to PDOs to be compliant to the EDS file standard.

Data location	Description	Used OD macro
Object dictionary	The object's actual data value is located within the context of the object dictionary. All accesses by the application only can be processed by the API functions of the CANopen stack.	OBD_SUBINDEX_RAM_VAR() or OBD_ROM_INDEX_RAM_ARRAY()
Application direct	The object's actual data value is located within the context of the application and is directly linked to a variable name. The control “Variable name” specifies the variable name in C notation. All accesses by the application can be processed directly by modifying the value of the variable or by the API function of the CANopen stack.	OBD_SUBINDEX_RAM_EXTVAR() or OBD_ROM_INDEX_RAM_EXTARRAY()
Application indirect	The object's actual data value is located within the context of the application and is indirectly linked to a variable name by using the VAR table. The control “Variable name” specifies the variable name in C notation. All accesses by the application can be processed directly by modifying the value of the variable or by the API function of the CANopen stack. For dynamic PDO mapping all objects must have this data location which are PDO mappable.	OBD_SUBINDEX_RAM_USERDEF() or OBD_ROM_INDEX_RAM_VARARRAY()

Table 13: Differences between the data locations

### 3.4.3.10 Sub-index Node Panel

Entries with object type DEFSTRUCT, RECORD or ARRAY contain sub-indices as child nodes. Sub-indices which are marked with a blue symbol  are exported via "Export" command. Sub-indices with a white symbol  are not exported.



The screenshot shows the 'Sub-index Node Panel' with the following controls:

- Export
- Sub index:
- Sub index name:
- Object type:
- Data type:
- Access type:
- Data location:
- Default Value:
- Use Value Range
- Lower Limit:
- Upper Limit:
- Variable name:
- Reserved string size:
- Write default value to EDS file
- Write default value to objdict.h
- PDO mappable
- Store to non-volatile memory
- Treat "ro" as "const"

Figure 37: Sub-index node panel

Note that the state of some controls of the sub-index node panel is changed dynamically. For example controls can be hidden or coloured grey (read-only). This state depends on the settings done before (e.g. data type, access type, ...).

Label	Control type	Description
Export	Check box	If the "Export" check box is checked the sub-index is exported with the "Export" command. Otherwise this object only exists as place holder (comparable with an out-commended function call in a C/C++ file). This check box is linked to the check box within the OD tree view at the selected sub-index node.
Sub index	Text box	Specifies the sub-index value within the range 0x00 to 0xFE.
Sub index name	Text box	Specifies the name of the sub-index.
Object type	Combo box	Specifies the object type of the sub-index. Possible values are: VAR, DEFTYPE or DOMAIN
Write default value to EDS file	Check box	Specifies whether the default value shall be written to the EDS file.
Write default value to objdict.h	Check box	Specifies whether the default value shall be written to the objdict.h file to be compiled with the CANopen stack. If this check box is not checked the OD macro in objdict.h is used with the suffix "_NOINIT". In this case the CANopen stack does not write any value on NMT event <i>Reset Node</i> or <i>Reset Communication</i> . Do not uncheck this check box for objects with access type = CONST!
PDO mappable	Check box	Specifies whether the sub-index can be mapped into a PDO. This control is only available for the object types VAR, DEFTYPE and for numeric data types.
Data type	Combo box	Specifies the data type of the object.
Access type	Combo box	Specifies the access type of the sub-index as written to the EDS file. Possible values are: <i>constant</i> , <i>read-only</i> , <i>read-write</i> , <i>read-write process input</i> , <i>read-write process output</i> , <i>write-only</i> . (refer to Table 12)
Data location	Combo box	Specifies the data location of the sub-index's actual data value. Refer to Table 13 for all possible values.
Store to non-volatile memory	Check box	Specifies whether the sub-index's actual data value shall be stored to the non-volatile memory if the "save" command is written to object 0x1010 (Store/Restore).
Treat "ro" as "const"	Check box	The access type of the sub-index is specified as to be written to the EDS file. All access types written to the EDS file have to be defined as defined within the CiA standards. But for the SYS TEC CANopen stack a "read-only" object can be treated as "const" object to reduce the RAM memory usage.
Default value	Text box	Specifies the default value for a DEFTYPE or VAR sub-index. The default value is set to the actual data value of the sub-index after initialization and after the NMT command <i>Reset Communication</i> and/or <i>Reset Node</i> .
Use Value Range	Check box	If this check box is checked the lower and upper limit can be defined for the data value of the sub-index.
Lower Limit	Text box	Specifies the lower limit for the data value of the sub-index.
Upper Limit	Text box	Specifies the upper limit for the data value of the sub-index.

Label	Control type	Description
Variable name	Text box	For sub-indices with data location “application direct” or “application indirect” (see Table 13) this text box specifies the variable provided by the application to which the sub-index is linked to. All OD accesses to the actual data value are redirected to the specified variable name (only the name but not the data type). The variable has to be specified in C notation.
Reserved string size	Numeric up/down	In case the data type of the sub-index is a string and the access type includes the write access rights then this value specifies the maximum length in characters (including the \0-termination) for the string held in RAM.

Table 14: Controls of sub-index node panel

### 3.4.4 Examples for creating objects manually

#### 3.4.4.1 Creating a VAR object index as numeric type

An OD entry of object type VAR does not have any sub-indices. The object type DEFTYPE is handled in the same manner as the object type VAR. Both object types can get a numeric data type, string type or a domain.

By adding a new object index by the context menu “Add new...” (see section 3.4.5) or by clicking the button “Add Index” within the object list node panel (see section 3.4.3.8) a VAR object is created as shown in Figure 38 (the value for the index is pre-filled). Firstly modify the object index value and the object name.

Make sure the checkbox “Export object” is checked. Otherwise the object will not be exported.

Figure 38: Example VAR object index as numeric type

With this example the result of the export command is the following:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt8,     kObdAccrW, tObdUnsigned8, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0 PDOMapping=0 ...</pre>

If changing the data type of the object to another numeric type then only changes the data types written to the export files:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt32,     kObdAccrW, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=rw DefaultValue=0 PDOMapping=0 ...</pre>

Changing the access type to “constant” results in the following export-output:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_ROM_VAR(0x2000, 0, kObdTypUInt32,     kObdAccrR, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=const DefaultValue=0 PDOMapping=0 ...</pre>

After changing the access type to “read-only” the checkbox “Tread ‘ro’ as ‘const’” becomes visible. If this checkbox is checked the same result is written to the file objdict.h as shown above, but the object gets the keyword “AccessType=ro” within the EDS file:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_ROM_VAR(0x2000, 0, kObdTypUInt32,     kObdAccrR, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=ro DefaultValue=0 PDOMapping=0 ...</pre>

If the checkbox “Tread ‘ro’ as ‘const’” keeps unchecked the OD macro OBD\_SUBINDEX\_RAM\_VAR is used with the access type kObdAccR:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt32, kObdAccR, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=ro DefaultValue=0 PDOMapping=0 ...                     </pre>

If changing the access type to “write-only” the exported files gets the following content:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0 PDOMapping=0 ...                     </pre>

If checking the checkbox “Index location in RAM” only the macro for beginning the index definition is changed within the file objdict.h – the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_RAM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0 PDOMapping=0 ...                     </pre>

Leaving the checkbox “Write default value to EDS file” unchecked, the keyword “DefaultValue” is not written to the EDS file – the file objdict.h keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_RAM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0 PDOMapping=0 ...                     </pre>



If unchecking the checkbox “Write default value to objdict.h”, the default value is not written to the file objdict.h– the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_RAM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR_NOINIT(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo PDOMapping=0 ...                     </pre>

If checking the checkbox “Use Value range” and filling out the “Lower Limit” and “Upper Limit” the value range is written to the export files, too. Note that this only can be done for numeric data types and if the checkboxes “Write default value to objdict.h” and “Write default value to EDS file” are checked:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_RAM(0x2000, 1, NULL) OBD_SUBINDEX_RAM_VAR_RG(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00000002, 0x00000001, 0x000001FF) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0x02 LowLimit=0x01 HighLimit=0x1FF PDOMapping=0 ...                     </pre>

If a “Callback handler” is specified for the object index (e.g. “AppManufacturerDataAccess”) it is written to the file objdict.h – the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_RAM(0x2000, 1, AppManufacturerDataAccess) OBD_SUBINDEX_RAM_VAR_RG(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, ManufacturerData, 0x00000002, 0x00000001, 0x000001FF) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0x02 LowLimit=0x01 HighLimit=0x1FF PDOMapping=0 ...                     </pre>

Additionally the prototype of the callback handler is written to the file obdcfg.h:

```

...
tCopKernel PUBLIC AppManufacturerDataAccess (CCM_DECL_INSTANCE_HDL_ tObdCbParam MEM* pParam_p);
...
    
```

If changing the “Data location” to “application direct” the textbox “Variable name” becomes visible and a variable name is proposed by the OD Builder II. The “Variable name” can be changed according to the needs of your application. As result the macro EXTVAR is used in file objdict.h – the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD BEGIN INDEX RAM(0x2000, 1, AppManufacturerDataAccess) OBD_SUBINDEX_RAM_EXTVAR_RG(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, dwManufacturerData_g, 0x00000002, 0x00000001, 0x000001FF) OBD_END_INDEX(0x2000) ... </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0x02 LowLimit=0x01 HighLimit=0x1FF PDOMapping=0 ... </pre>

Additionally the prototype of the external variable is written to the file obdcfg.h:

```

...
extern tObdUnsigned32 MEM dwManufacturerData_g;
...

```

If changing the “Data location” to “application indirect” the textbox “Variable name” keeps visible including a variable name for the application. The “Variable name” can be changed according to the needs of your application. As result the macro USERDEF is used in file objdict.h – the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD BEGIN INDEX RAM(0x2000, 1, AppManufacturerDataAccess) OBD_SUBINDEX_RAM_USERDEF_RG(0x2000, 0, kObdTypUInt32, kObdAccW, tObdUnsigned32, dwManufacturerData_g, 0x00000002, 0x00000001, 0x000001FF) OBD_END_INDEX(0x2000) ... </pre>	<pre> ... [2000] ParameterName=ManufacturerData ObjectType=0x07 DataType=0x0007 AccessType=wo DefaultValue=0x02 LowLimit=0x01 HighLimit=0x1FF PDOMapping=0 ... </pre>

There is no prototype for the external variable in obdcfg.h but an entry is written to the VAR table within the file vartab.h:

```

...
extern tObdUnsigned32 MEM dwManufacturerData_g;
...
CONST tVarParam ROM aVarTab_g[] =
{
    {kVarValidAll, 0x2000, 0x00, sizeof(dwManufacturerData_g), &dwManufacturerData_g,
    NULL, NULL},
    ...
};
...

```

### 3.4.4.2 Creating a VAR object index as string type

A new object index must be created within the OD Builder II as shown in Figure 38 (no further changes). The data type has to be changed to a string type (e.g. VISBLE\_STRING). The resulting Editor Panel is shown in Figure 39.

**Object Index 0x2000**  Export object

Object index:  Data type:  Default Value:

Object name:  Access type:  Reserved string size:

Callback handler:

Object type:

Category:

Index location in RAM  Store to non-volatile memory

Write default value to EDS file

Write default value to objdict.h

Figure 39: Example VAR object index as string type

The following controls are hidden:

- PDO mappable
- Data location
- Use Value Range
- Lower Limit
- Upper Limit

Additionally the control “Reserved string size” is shown. Here the maximum string length is specified. The CANopen stack reserves this amount of memory in the RAM for this string object.

With this example the result of the export command is the following:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_RAM_VSTRING(0x2000, 0, kObdAccRW,     ManufacturerString, 32, "0") OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerString ObjectType=0x07 DataType=0x0009 AccessType=rw DefaultValue=0 ...</pre>

If changing the access type to “constant” the control “Reserved string size” becomes hidden and the content of the exported files changes as followed:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_ROM_VSTRING(0x2000, 0, kObdAccR,     ManufacturerString, 32, "0") OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerString ObjectType=0x07 DataType=0x0007 AccessType=const DefaultValue=0 ...</pre>

The “Callback handler” also can be specified for object indexes with string type as shown in section 3.4.4.1.

### 3.4.4.3 Creating a DOMAIN object index

After creating a new index as shown in Figure 38 (no further changes) the data type has to be changed to DOMAIN. The data type DOMAIN can be set for the object types VAR, DEFTYPE and

DOMAIN. If the object type DOMAIN is selected, only DOMAIN can be selected for the data type. The resulting Editor Panel is shown in Figure 40.

Figure 40: Example DOMAIN object index

With this example the result of the export command is the following:

File objdict.h	EDS file
<pre>... OBD_BEGIN_INDEX_ROM(0x2000, 1, NULL)   OBD_SUBINDEX_RAM_DOMAIN(0x2000, 0, kObdAccrW,     ManufacturerDomain) OBD_END_INDEX(0x2000) ...</pre>	<pre>... [2000] ParameterName=ManufacturerDomain ObjectType=0x02 DataType=0x000F AccessType=rw ...</pre>

There is an entry written to the VAR table for DOMAN objects within the file vartab.h:

```
...
extern BYTE MEM abManufacturerDomain_g[];
...
CONST tVarParam ROM aVarTab_g[] =
{
  {kVarValidAll, 0x2000, 0x00, sizeof(abManufacturerDomain_g), &abManufacturerDomain_g[0],
    NULL, NULL},
  ...
};
...
```

### 3.4.4.4 Creating an ARRAY object index

An object of type ARRAY always has a numeric data type. It consists of sub-indices whereat the sub-index 0 holds the number of entries. The ARRAY object can contain a maximum of 254 entries (not included the sub-index 0). All entries beginning from sub-index 1 have the same data type, access type and default value as long as the same data location.

The SYS TEC CANopen stack supplies a special macro for ARRAY objects. This special macro reduces the ROM usage but the usage of RAM increases slightly. It is used if the global setting “Use ARRAY macros in OD” is checked within the project node panel (see section 3.4.3.1) AND if the checkbox “Use ARRAY macro for object index” is checked. If the checkbox “Use ARRAY macros in OD” is unchecked then ALL the ARRAY objects are written to the file objdict.h one by one (each sub-index is created separately). But if only the checkbox “Use ARRAY macro for object index” is unchecked only the appropriate ARRAY object is written to the file objdict.h with each sub-index separately.

Based on creating a new object as shown in Figure 38 the “Object type” has to be changed to “ARRAY”. The resulting Editor Panel is shown in Figure 41. Firstly specify the “Number of subindices”. The lowest value is 2 and the highest value is 255. This value includes the sub-index 0!

**Object Index 0x2000**  Export object

Object index:  Data type:  Default Value:

Object name:  Access type:   Use Value Range

Callback handler:  Data location:  Lower Limit:

Object type:  Number of subindices:  Upper Limit:

Category:

Index location in RAM  Use ARRAY macro for object index

Write default value to EDS file  Sub index 0x00 is read-write

Write default value to objdict.h  Store to non-volatile memory

PDO mappable

Figure 41: Example ARRAY object index

Using this example the result of the export command is the following:

File objdict.h	EDS file
<pre> ... OBD_ROM_INDEX_RAM_ARRAY(0x2000, 1, NULL, kObdTypUInt8,     kObdAccrW, tObdUnsigned8, ManufacturerArray, 0x00) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerArray ObjectType=0x08 SubNumber=2  [2000sub0] ParameterName=Number of entries ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=1 PDOMapping=0  [2000sub1] ParameterName=array sub index ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0x0 PDOMapping=0 ...                     </pre>

Similar to the description in section 3.4.4.1 the “*Data location*” can be changed for ARRAY objects. If selecting the “*Data location*” as “*application direct*” the EXTARRAY macro is used and a “*Variable name*” is proposed by the OD Builder II without brackets. The EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_ROM_INDEX_RAM_EXTARRAY(0x2000, 1, NULL, kObdTypUInt8,     kObdAccRW, tObdUnsigned8, abManufacturerArray_g, 0x00) ... </pre>	<pre> ... [2000] ParameterName=ManufacturerArray ObjectType=0x08 SubNumber=2  [2000sub0] ParameterName=Number of entries ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=1 PDOMapping=0  [2000sub1] ParameterName=array sub index ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0x0 PDOMapping=0 ... </pre>

Additionally the prototype of the external variable is written to the file obdcfg.h:

```

...
extern tObdUnsigned8    MEM abManufacturerArray_g[];
...

```

If “*application indirect*” is selected as “*Data location*” the VARARRAY macro is used and a “*Variable name*” is proposed by the OD Builder II without brackets. The EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_ROM_INDEX_RAM_VARARRAY(0x2000, 1, NULL, kObdTypUInt8,     kObdAccRW, tObdUnsigned8, abManufacturerArray_g, 0x00) ... </pre>	<pre> ... [2000] ParameterName=ManufacturerArray ObjectType=0x08 SubNumber=2  [2000sub0] ParameterName=Number of entries ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=1 PDOMapping=0  [2000sub1] ParameterName=array sub index ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0x0 PDOMapping=0 ... </pre>

There is no prototype for the external variable in obdcfg.h but an entry is written to the VAR table within the file vartab.h:

```

...
extern tObdUnsigned8    MEM abManufacturerArray_g[];
...
CONST tVarParam ROM aVarTab_g[] =
{
    {kVarValidAll, 0x2000, 0x01, sizeof(abManufacturerArray_g[0]), &abManufacturerArray_g[0],
        NULL, NULL},
    ...
};
...

```

The “*Callback handler*” and the “*Access type*” also can be specified for ARRAY objects as described in section 3.4.4.1. But note that an ARRAY object cannot be specified as “*constant*” for the SYS TEC CANopen stack. The result will be an object with access type “*read-only*” within the file objdict.h. In this case the Output Window shows a warning:

```

...
Warning: An ARRAY is defined as const object (index 0x2000).
...

```

After selecting a sub-index of the ARRAY object within the project tree, the Editor Panel shows all settings of this entry. They are coloured grey (disabled) except “*Sub index name*”. All disabled parameters are copied from the settings within the object index node panel for ARRAY objects and cannot be changed within the sub-index node panel.

You can specify another name for the sub-indices beginning from sub-index 1. As result only the name of the sub-index is changed within the EDS file:

File objdict.h	EDS file
<pre> ... OBD_ROM_INDEX_RAM_VARARRAY(0x2000, 1, NULL, kObdTypUInt8,     kObdAccRW, tObdUnsigned8, abManufacturerArray_g, 0x00) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerArray ObjectType=0x08 SubNumber=2  [2000sub0] ParameterName=Number of entries ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=1 PDOMapping=0  [2000sub1] ParameterName=Changed name ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0x0 PDOMapping=0 ...                     </pre>

After unchecking the checkbox “Use ARRAY macro for object index” all sub-indices are written separately to the file objdict.h – the EDS file keeps unchanged:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_ROM(0x2000, 2, NULL) OBD_SUBINDEX_ROM_VAR(0x2000, 0, kObdTypUInt8, kObdAccR,     tObdUnsigned8, Number_of_entries, 0x01) OBD_SUBINDEX_RAM_VAR(0x2000, 1, kObdTypUInt8, kObdAccRW,     tObdUnsigned8, Changed_name_for_sub_index, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerArray ObjectType=0x08 SubNumber=2  [2000sub0] ParameterName=Number of entries ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=1 PDOMapping=0  [2000sub1] ParameterName=Changed name ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0x0 PDOMapping=0 ...                     </pre>



### 3.4.4.5 Creating an object index as RECORD

An object of type RECORD does not have any sub-indices, but in contrast to ARRAY objects all sub-indices may have different data types, access types and/or default values. The object type DEFSTRUCT is handled in the same manner as the object type RECORD. Each RECORD can have up to 255 sub-indices (including sub-index 0). Usually the sub-index 0 defines the “number of entries” or “highest sub-index supported”.

Based on creating a new object as shown in Figure 38 the “Object type” has to be changed to “RECORD”. The resulting Editor Panel is shown in Figure 42. The OD Builder II automatically creates the first two sub-indices.

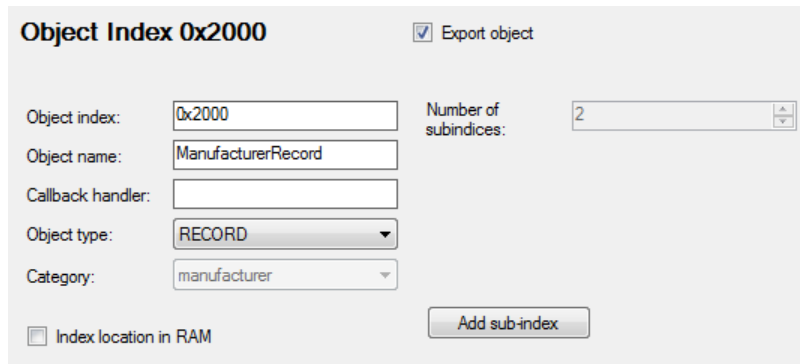


Figure 42: Example RECORD object index

Using this example the result of the export command is the following:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_ROM(0x2000, 2, NULL)   OBD_SUBINDEX_ROM_VAR(0x2000, 0, kObdTypUInt8, kObdAccR,     tObdUnsigned8, Highest_sub_index_supported, 0x01)   OBD_SUBINDEX_RAM_VAR(0x2000, 1, kObdTypUInt8, kObdAccRW,     tObdUnsigned8, sub_index_1, 0x00) OBD_END_INDEX(0x2000) ...                     </pre>	<pre> ... [2000] ParameterName=ManufacturerRecord ObjectType=0x09 SubNumber=2  [2000sub0] ParameterName=Highest sub-index supported ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=0x01 PDOMapping=0  [2000sub1] ParameterName=sub-index 1 ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0 PDOMapping=0 ...                     </pre>

In contrast to ARRAY objects the sub-indices of RECORD objects can be fully edited within the Editor Panel.

How to add a new sub-index to a RECORD object or change an existing one is described in section 3.4.4.6.

### 3.4.4.6 Creating a new sub-index

For adding a new sub-index to a RECORD object index, click the button “Add sub-index” within the Editor Panel for a selected RECORD object (see Figure 42). A new sub-index is automatically added at the next free sub-index value as a copy of the previous one.

To adapt the properties of the newly created sub-index, select it within the project tree – the right Editor Panel changes as shown in Figure 43 – and firstly specify a “Sub index name” for the new sub-index.

Figure 43: Example sub-index

Only the object types “VAR”, “DEFTYPE” or “DOMAIN” can be assigned to a sub-index.

In opposition to the Editor Panel that is shown for an object index of type numeric VAR (see Figure 38) some parameters cannot be specified for sub-indices here. The controls “Callback handler”, “Category” and “Index location in RAM” only can be specified for the parent node (the object index).

The result is a new sub-index entry within the file objdict.h and within the EDS file:

File objdict.h	EDS file
<pre> ... OBD_BEGIN_INDEX_ROM(0x2000, 3, NULL)   OBD_SUBINDEX_ROM_VAR(0x2000, 0, kObdTypUInt8, kObdAccR,     tObdUnsigned8, Highest_sub_index_supported, 0x02)   OBD_SUBINDEX_RAM_VAR(0x2000, 1, kObdTypUInt8, kObdAccRW,     tObdUnsigned8, sub_index 1, 0x00)   OBD SUBINDEX RAM VAR(0x2000, 2, kObdTypUInt8, kObdAccRW,     tObdUnsigned8, NewSubindex, 0x00) OBD_END_INDEX(0x2000) ... </pre>	<pre> ... [2000] ParameterName=ManufacturerRecord ObjectType=0x09 SubNumber=3  [2000sub0] ParameterName=Highest sub-index supported ObjectType=0x07 DataType=0x0005 AccessType=const DefaultValue=0x02 PDOMapping=0  [2000sub1] ParameterName=sub-index 1 ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0 PDOMapping=0  [2000sub2] ParameterName=NewSubindex ObjectType=0x07 DataType=0x0005 AccessType=rw DefaultValue=0 PDOMapping=0 ... </pre>

All other adaptations that can be done for sub-indices are similar to the ones that are described in section 3.4.4.1, 3.4.4.2 and 3.4.4.3. But the changes are only applied to the appropriate sub-index they were made for.

### 3.4.5 Context Menu

A context menu is opened after right clicking on a node within the project tree (see Figure 44). Depending on the node-type different commands are available. E.g. only index nodes and sub-index nodes can be copied.

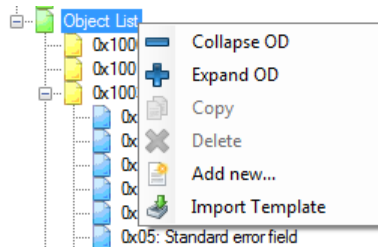


Figure 44: Context menu

Select “*Collapse OD*” to collapse all object index nodes. When clicking to “*Expand OD*” all object index nodes are expanded.

To copy an object index or sub-index, the appropriate object index or sub-index has to be selected first. Use the context menu command “*Copy*”. The new object index or sub-index is inserted automatically at the next free index or sub-index. The properties of the pasted object index or sub-index can be changed within the Editor Panel on the right.

If an object index or sub-index is not needed anymore it can be deleted by the use of the context menu command “*Delete*”.

The “*Add new...*” command creates a new object index or sub-index. If the “*Object List*” node or an object index node is selected, the “*Add new...*” command adds a new object index (see section 3.4.4.1 and the following). If a sub-index is selected a new sub-index is added (see section 3.4.4.6).

Via the “*Import Template*” command a prepared template file can be inserted into the object list. Refer to section 3.4.6 for detailed information.

### 3.4.6 Import templates

With the installation of the OD Builder II tool several template files are installed to the folder %InstallDir%\Knowledge\. These template files include one or more predefined object indexes and can be used to add prepared object indexes to an existing project. Furthermore template files can be used to overwrite settings of an existing project and to reset the included object indexes to their default settings defined within the template files. Table 15 lists all available template files. In future releases of OD Builder II there might be more template files installed than listed here.

Template file	CiA standard	Description
CiA-301_generic.odbtpl	CiA-301	Includes all known objects of CiA-301 including 2 SDO servers, 1 SDO client, 4 RPDOs and 4 TPDOs. This template is used to create a new CANopen Slave or Master project.
CiA-301_4RPDO_gran8.odbtpl	CiA-301	Includes 4 receive PDOs with a PDO granularity of 8 bits.
CiA-301_4TPDO_gran8.odbtpl	CiA-301	Includes 4 transmit PDOs with a PDO granularity of 8 bits.
CiA-301_2SDOS.odbtpl	CiA-301	Includes 2 SDO server entries.
CiA-301_1SDOC.odbtpl	CiA-301	Includes 1 SDO client entries.

Table 15: Available template files

To import a template file use the context menu “*Import Template*” (see section 3.4.5) or the button “*Import Template*” within the object list node panel (see section 3.4.3.8).

After selecting the template file name via the file browser the OD Builder II checks whether the included object indexes already exist within the project. If there are any existing object indexes, the OD Builder II asks if they shall be overwritten (see Figure 45). Check “apply to following” if your decision shall be used for all existing object indexes. Otherwise this dialog box is shown for every object index included in the template file that already exists within the project.

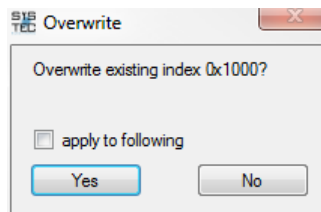


Figure 45: Overwrite an existing object index

If the template file implements a device profile (the object indexes are starting from index 0x6000) it can be added with an offset of a multiple of 0x0800. This feature can be used to realize virtual CANopen devices. OD Builder II opens a dialog for editing the import offset as shown in Figure 46. Use the up/down controls to change the offset value. The offset value is shown hexadecimal and only can be incremented or decremented in multiples of 0x0800.

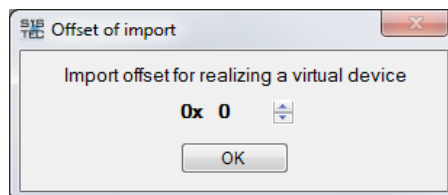


Figure 46: Import offset dialog

After clicking to the OK button the OD Builder II imports all included object indexes with the configured offset value.

**Example:**

- The template file includes the object indexes 0x6000 and 0x6102.
- The import offset is set to 0x0800.
- Result: The object indexes 0x6800 and 0x6902 are imported.

### 3.5 Verify a project

A project can be verified by clicking “File / Verify”.

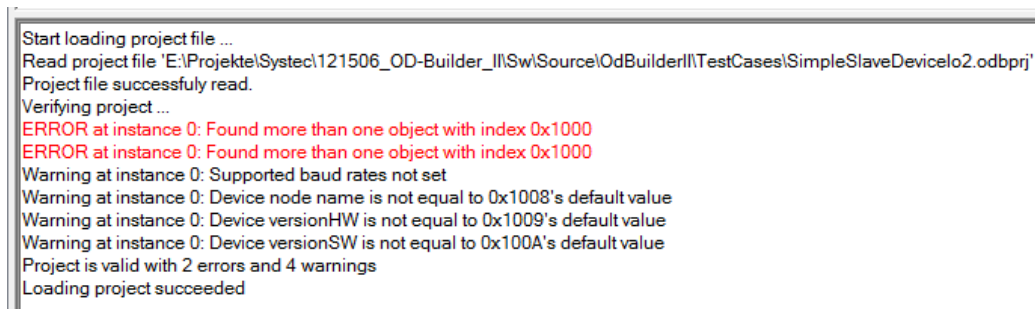
The verify command checks all settings within the project whether they are consistent and whether they fulfil the CiA standards. Settings of the object dictionary are verified by using a Data Base File:

```
%InstallDir%\Knowledge\CANopen_generic.odbbas
```

This Data Base File is created by SYS TEC electronic GmbH and used by the OD Builder II to check your project against CiA-301 standard version V4.2.0.72. **Do not change this file!**

Additionally the OD Builder II verifies a project automatically after loading (see section 3.2), after importing an EDS file (see section 3.1.2) and before exporting the object dictionary (see section 3.6).

If there are warnings and/or errors the OD Builder II displays a Dialog Window as shown in Figure 19. Choose “Yes” or “No” whether to continue the loading/exporting or not. Additionally the lower Output Window (see Figure 47) lists all warnings and errors. Errors are displayed in red colour.



```
Start loading project file ...
Read project file 'E:\Projekte\System\121506_OD-Builder_II\Sw\Source\OdBuilder\ITestCases\SimpleSlaveDeviceLo2.odbpj'
Project file successfully read.
Verifying project ...
ERROR at instance 0: Found more than one object with index 0x1000
ERROR at instance 0: Found more than one object with index 0x1000
Warning at instance 0: Supported baud rates not set
Warning at instance 0: Device node name is not equal to 0x1008's default value
Warning at instance 0: Device versionHW is not equal to 0x1009's default value
Warning at instance 0: Device versionSW is not equal to 0x100A's default value
Project is valid with 2 errors and 4 warnings
Loading project succeeded
```

Figure 47: Output Window showing the verification result

Before exporting a project, all errors must be fixed. Even though there are warnings, the project can be exported. But there could be problems on compile-time or run-time of the CANopen stack or the EDS file could be not conform to the CiA standard.

### 3.6 Export a project

Before a project can be exported the project needs to be saved (refer to section 3.3).

**Note:**

Only the objects are exported to another file format which “Export” checkbox is checked (refer to sections 3.4.3.9 and 3.4.3.10).

To export a project click to the menu command “Export” as shown in Figure 48.

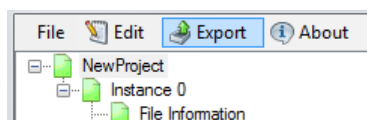


Figure 48: Menu command Export

If the project is not saved yet, the OD Builder II asks for saving the project as shown in Figure 49.

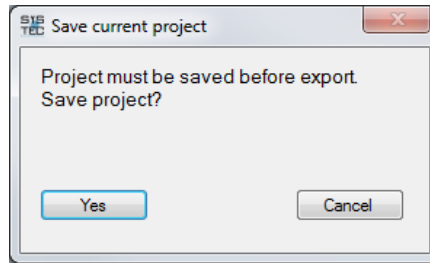


Figure 49: Save before export

If the project is saved the Export dialog window opens with settings for the export process (see Figure 50).

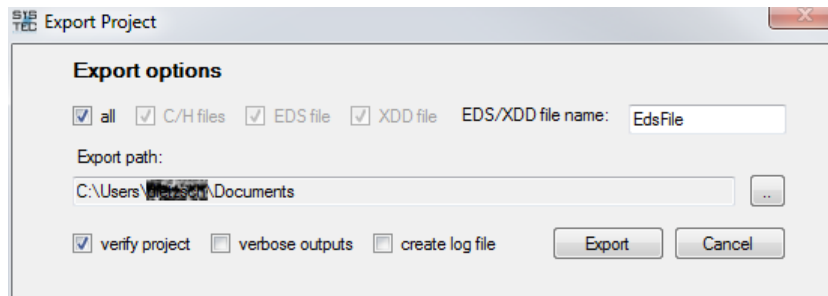



Figure 50: Export dialog window

On the top left the files that shall be exported can be chosen. Table 16 lists all format types. The file name used for the EDS file is specified within the “EDS/XDD file name” text box. All files will be exported to the path specified within the “Export path” text box. The export path can be changed by clicking to the browse button on the right side: 

Export format type	Description
all	All export files are written. If this option is chosen then the other checkboxes are checked and disabled automatically.
C/H file	All *.c/*.h files to be compiled with the SYS TEC CANopen stack are exported.
EDS file	An EDS file is written according to the CiA-306 standard. The file name to be used is specified within the “EDS/XDD file name” text box.
XDD file	An XDD file is written according to the CiA-311 standard. The file name to be used is specified within the “EDS/XDD file name” text box. This format is not supported in the current version of the OD Builder II.

Table 16: Export format types

Check the checkbox “verify project” if the project shall be verified before exporting. If the checkbox “verbose outputs” is checked the Output Window displays detailed information while the export process is running. Otherwise only warnings and/or errors are shown. If the checkbox “create log file” is checked then all outputs of the export process are written to a log file. The log file is written to the export path specified above. The log file name is fixed to “ODBuilderII.log”.

Figure 51 shows a typical export result exporting all formats but without verbose mode.

```

Start exporting project
Start export of instance 0 ...
ODBuilder II - V1.0.0.1 - SYS TEC electronic GmbH
Read project file 'E:\Projekt\Systemc\121506_OD-Builder_II\Sw\Source\OdBuilder\IITestCases\d
Project file successfully read.
Writing 'EdsFile.edb' ...
EDS file successfully written.
The output type 'xdd' is currently not implemented!
Writing 'objdict.h' ...
Writing 'objdict.c' ...
Writing 'obdcfg.h' ...
Writing 'vartab.h' ...
Object dictionary files successfully written.
The output type 'cfg' is currently not implemented!
Export of instance 0 finished
    
```

Figure 51: Output Window showing the export results

**Note:**

Exporting an OD Builder II project to an EDS file some information of the project gets lost (e.g. the callback handler, name of the variable, ...). If an exported EDS file is imported again, the new resulting project differs to the original one. For this reason always use the OD Builder II project file as base for all operations and changings.

**3.7 Command line arguments**

The frontend (GUI) of the OD Builder II is displayed with the executable file %InstallDir%\Bin\OdbGui.exe. This executable file can be used with command line arguments. Use the following command line format:

```
OdbGui.exe [<project_file>] [options]
```

When a valid OD Builder II project filename is specified, the OD Builder II tool is started and the specified project file is opened automatically.

**Example 1:**

```
OdbGui.exe C:\systemc\cop\objdicts\ds301_3p\ds301_3p.odbprj
```

Table 17 lists all available command line options of the OdbGui.exe.

Option	Description
--verbose	Displays additional information within the output window for debug purposes.

Table 17: Command line options

**Example 2:**

```
OdbGui.exe --verbose
```

## 4 Known issues

- The OD Builder II does not support ARRAY objects including sub-indices of data type DOMAIN, VISIBLE\_STRING, OCTET\_STRING and/or UNICODE\_STRING. Use the object type RECORD instead of ARRAY if you need any of these data types.
- The OD Builder II does not support different access types for sub-indices of an ARRAY object.

**Note:**

If there are undocumented issues, please activate the verbose output mode of the OD Builder II tool. Refer to the sections 3.6 and/or 3.7 for more information how to activate the verbose mode. Repeat all steps which results to the issue and copy the content of the output window. Additionally make screenshots and send them with a detailed description of the issue to the support email address [support@systec-electronic.com](mailto:support@systec-electronic.com).



## Index

### A

About .....	15
Add Index .....	31
Add Instance .....	22
Add sub-index .....	34, 50

### C

colour.....	21
Command line .....	55
Context menu .....	51
Copy .....	51

### D

Data location	
Application direct .....	35, 42
Application indirect .....	35, 42
Object dictionary .....	35
Debug .....	55

### E

Examples.....	38
Export .....	53
extension .....	17

### I

Import	
EDS .....	17
Template .....	31
templates .....	51
Installation .....	9
Instance number.....	23
Introduction.....	7

### L

Load .....	19
------------	----

### N

NewProject .....	16
Number of indexes .....	31

### O

Object type	
ARRAY .....	44
DEFSTRUCT.....	49
DEFTYPE .....	38
DOMAIN.....	43
RECORD .....	49
VAR .....	38
Overview .....	7

### P

Panel	
Comments Node .....	30
Device Abilities Node.....	27
Device Information Node .....	25
Dummy Usage Node .....	29
File Information Node .....	24

Instance Node .....	23
Object List Node.....	31
Object Node .....	32
Project Node .....	22
Sub-index Node .....	36
Parameter	
Access type.....	33, 37, 39
Act. Baud rate .....	29
BaudRate .....	28
Boot-up master.....	28
Boot-up slave .....	28
Build date.....	26
Callback handler.....	33, 41
CANopen Manager.....	29
Category .....	33
Comments.....	31
Configuration manager .....	28
Created by .....	25
Creation date.....	25
Creation time.....	25
Data location .....	33, 37, 42
Data type .....	33, 37, 39
Default value .....	34, 37
Device family.....	26
Dummy usage .....	29
EDS file name .....	25
EDS file version.....	25
EDS version .....	25
Export .....	33, 37
Firmware version.....	26
Flying master.....	28
Format name.....	25
Hardware version .....	26
Index location in RAM.....	33, 40
Lower Limit.....	34, 37
LSS Master Support .....	28
LSS Slave Support .....	28
Modification date .....	25
Modification time .....	25
Modified by.....	25
Network name.....	29
Network num.....	29
Node ID.....	23, 29
Node name .....	29
Num Rx PDO .....	28
Num Tx PDO.....	28
Number of subindices.....	34
Object index .....	33
Object name.....	33
Object type.....	33, 37
OD file name .....	23
OD init function name .....	23
Order number.....	26
PDO Granularity.....	28
PDO mappable.....	33, 37
Product ID.....	26
Product name.....	26
Profile technology.....	22
Project Name .....	22
Reserved string size .....	34, 38, 43
SDO client.....	28
SDO manager .....	28
Self Start Device.....	28

Software version..... 26  
Specification revision..... 26  
Store to non-volatile memory ..... 34, 37  
Sub index ..... 37  
Sub index 0x00 is read-write..... 34  
Sub index name ..... 37  
Support MPDO ..... 28  
Treat ro as const ..... 34, 37, 39  
Upper Limit..... 34, 37  
Use ARRAY macro for object index ..... 34, 44, 48  
Use ARRAY macro in OD ..... 22  
Use ARRAY macros in OD ..... 44  
Use Value Range ..... 34, 37, 41  
VAR table name ..... 23  
Variable name ..... 34, 38  
Vendor ID ..... 26  
Vendor name..... 26  
Write default value to EDS file ..... 33, 37, 40  
Write default value to objdict.h ..... 33, 37, 41

**R**

Recently used ..... 19  
Redo ..... 21

**S**

Save ..... 16, 20  
Save As..... 20

**U**

Undo ..... 21

**V**

Verbose..... 54, 55  
Verify..... 53

**Document:** Quick-start manual OD Builder II  
**Document Number:** L-1852e\_01, Edition Mai 2016

---

**Do you have any suggestions for improving this manual?**

---

---

---

---

**Have you found any mistakes in this manual?**

**Page**

---

---

---

---

**Sent from:**

Customer number: \_\_\_\_\_

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

---

**Send to:** SYS TEC electronic GmbH  
Am Windrad 2  
D - 08468 Heinsdorfergrund  
GERMANY  
Fax: +49 (0) 37 65 / 38600-4100  
Email: [info@systec-electronic.com](mailto:info@systec-electronic.com)

