User's guide

CARS Builder

Version: 1.3
Date: Mai 2003

YP Telemetrie bv

Contents

1 Introduction

CARS 6 is a telemetry system with advanced Supervisory Control and Data Acquisition (SCADA) functionality designed for stand-alone systems for a variety of industries.
CARS Builder is an powerful and intuitive application generator. This means that all the control and monitoring facilities are already built into the system, and only the project definitions need to be provided by the application engineer. Minimal computer or programming skills are required.
It is an easy to use tool to build a dynamic presentation in a minimum of time.
Presentation of the background, and dynamic components, can be done by each graphic program which can make a .bmp extension. So the engineer can use an experienced program and don’t need to learn working with a new tool.

Some useful features are:

- background images import from any graphic design program,
- dynamic components library available,
- components library configurable and extendible by users,
- database connectivity,
- development in native language and runtime in any language,
- on-line testing,
- re-use of existing presentations to a new application by templates,
- advanced control algorithms possibility,
- automatic connection to CARS 6 maintenance software.

1.1What’s in this manual

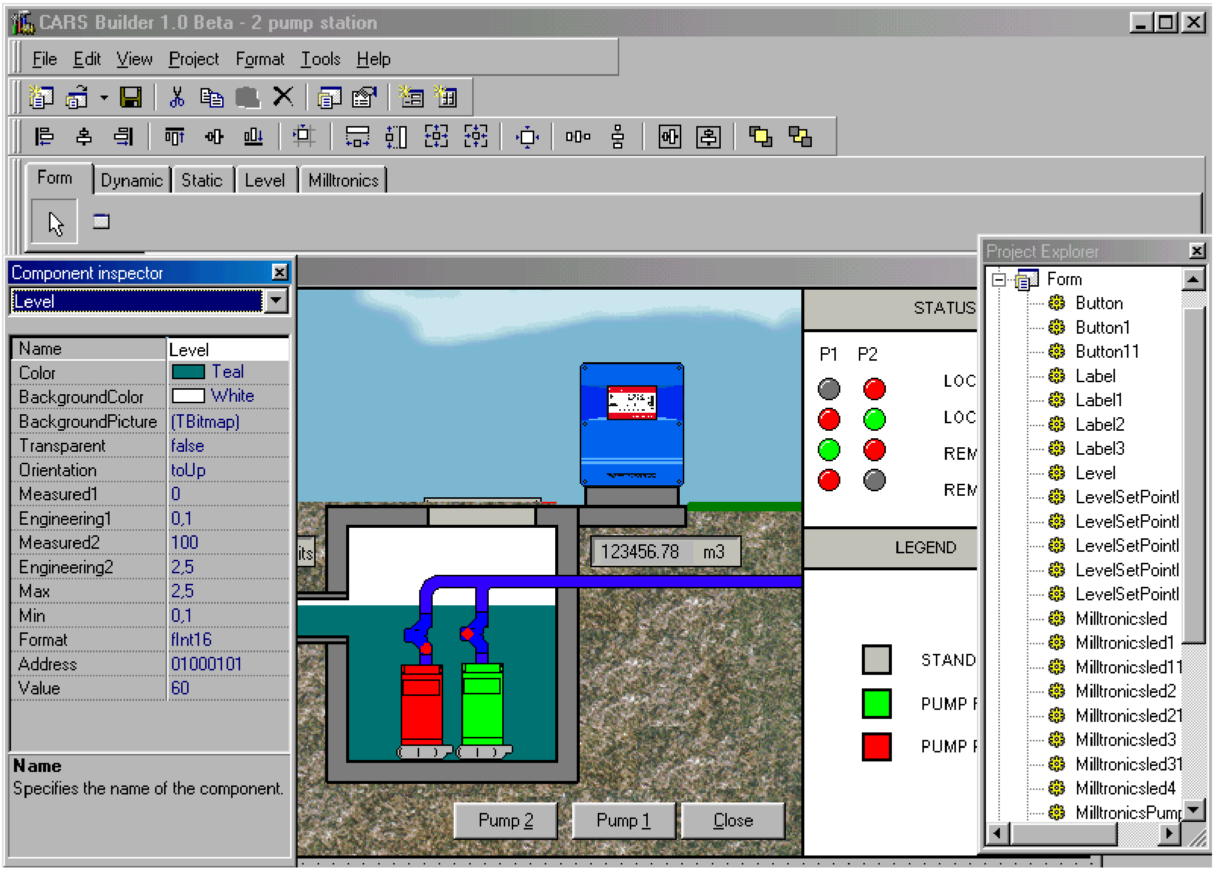
In this manual is described how to use the CARS Builder working environment and the tools it provides.
The User guide is so organised that at first the development enviroment will be described. Then you will get a tour through the CARS Builder, creating a simple application, and later on the basics of using the CARS Builder and his components will be more explained.

2 The development environment

When you start CARS Builder, you are immediately placed within the visual development environment. It is within this environment that CARS Builder provides all the tools you need to design, develop and test CARS 6 station applications. This chapter describes the development environment and the tools that are available in the CARS Builder environment.

2.1Main window

The CARS Builder development environment has several flexible parts that you can locate on the screen. The Main window contains the main menu, toolbars, Form designer and Component palette.The Component inspector and a form are automatically displayed. As you are working, you can re-size each part and display additional tools as needed.



Name	Level
Color	Teal
BackgroundColor	White
BackgroundPicture	(TBitmap)
Transparent	false
Orientation	toUp
Measured1	0
Engineering1	0,1
Measured2	100
Engineering2	2,5
Max	2,5
Min	0,1
Format	fInt16
Address	01000101
Value	60

Name
Specifies the name of the component.

Figure 2.1: CARS Builder development environment.

2.2Menus and toolbars

The main menu at the top of the Main window provides access to the commands and tools in the CARS Builder development environment. Toolbars are located under the main menu. The toolbars have buttons that provide quick access to frequently used operations and commands. All toolbar commands are duplicated in the main menu. To find out what a button does, you can point with the mouse to it and a hint is displayed.

Context menus are available when you right-click on many elements in the CARS Builder development environment. A context menu displays commands that relate to the object you are working with. Context menus are available for: Forms, Components, Component Palette, Component inspector, Project explorer and toolbars.

Menu	Menu command	Description
File	New Project…	Opens the New Project dialog box which contains project templates.
	Open Project…	Opens an existing project.
	Close Project	Closes the currently open project.
	Save Project	Saves changes made in the project to the project file.
	Save Project As…	Saves the project to a new name or location.
	Recent Projects	Reopens a recently saved or closed project.
	Exit	Closes the open project and then closes CARS Builder.

Table 2.1: File menu commands.

Menu	Menu command	Description
Edit	Cut	Removes components from the active form and place them on the Clipboard.
	Copy	Places an exact copy of the selected component on the Clipboard.
	Paste	Inserts the contents of the Clipboard into the active form.
	Delete	Removes the selected component without placing a copy on the Clipboard.
	Select All	Selects every component on the active form.

Table 2.2: Edit menu commands.

Menu	Menu command	Description
View	Project Explorer	Displays the Project Explorer. If the Project Explorer is already open, it becomes the active window.
	Component inspector	Displays the Component inspector. If the Component inspector is already open, it becomes the active window.
	Toolbars - Component Palette	Shows or hides the Component Palette.
	Toolbars - Layout	Shows or hides the Layout toolbar.
	Toolbars - Standard	Shows or hides the Standard toolbar.

Table 2.3: View menu commands.

Menu	Menu command	Description
Project	Add Form…	Opens the New Form dialog box which contains form templates.
	Add Component…	Opens the New Component dialog box which contains component templates.
	Create - Project Template…	Opens the Create Project Template dialog box to create a template for the current project.
	Create - Form Template…	Opens the Create Form Template dialog box to create a template for the active form.
	Create - Component Template…	Opens the Create Component Template dialog box to create a template for the active component.
	Logging...	Opens the Logging dialog box to create and edit log messages.
	Cross Reference...	Opens the cross reference dialog box to have an overview of the components used in the project.
	Language...	Opens the Language dialog box to create and edit additional languages.
	Options...	Opens the Project Options dialog box for setting project options.

Table 2.4: Project menu commands.

Menu	Menu command	Description
Format	Align - Lefts	Aligns the selected components to the left edge of the component first selected.
	Align - Centers	Moves the selected components horizontally until their centers are aligned with the component first selected.
	Align - Rights	Aligns the selected components to the right edge of the component first selected.
	Align - Tops	Aligns the selected components to the top edge of the component first selected.
	Align - Middles	Moves the selected components vertically until their centers are aligned with component first selected.
	Align - Bottoms	Aligns the selected components to the bottom edge of the component first selected.
	Align - To Grid	Aligns the selected components to the grid.
	Make Same Size - Width	Resizes the selected components to the width of the component first selected.
	Make Same Size - Height	Resizes the selected components to the height of the component first selected.
	Make Same Size – Width and Height	Resizes the selected components to the width and height of the component first selected.
	Make Same Size - Size To Grid	Resizes the selected components to the grid.
	Horizontal Spacing - Make Equal	Horizontally aligns three or more selected components so that the middle components are equidistantly spaced between the outer components.
	Vertical Spacing - Make Equal	Vertically aligns three or more selected components so that the middle components are equidistantly spaced between the outer components.
	Center in Form - Horizontally	Horizontally lines up the selected components with the center of the form.
	Center in Form - Vertically	Vertically lines up the selected components with the center of the form.
	Order - Bring to Front	Moves a selected component in front of all other components on the form.
	Order - Send to Back	Moves a selected component behind all other components on the form.
	Lock Controls	Secures all components on the active form in their current position.

Table 2.5: Format menu commands.

Menu	Menu command	Description
Tools	External Tools…	Opens the External Tools dialog box. Use this dialog box to add, delete, or edit programs on the Tools menu.
	Customize - Project Library…	Opens the Customize Project Library dialog box to manage project templates.
	Customize - Form Library…	Opens the Customize Form Library dialog box to manage form templates.
	Customize - Component Library…	Opens the Customize Component Library dialog box to manage component templates.
	Customize - Component Palette…	Opens the Customize Component Palette dialog box for customizing the way the component palette appears.
	Import Library…	Installs new components into CARS Builder.
	Options…	Opens the Options dialog box. Use this dialog box to customize the development environment.

Table 2.6: Tools menu commands.

Menu	Menu command	Description
Help	Contents and index…	Displays the Help Topics dialog box.
	CARS Builder on the Web	Opens the web browser and displays the CARS Builder home page.
	About CARS Builder	Displays the About CARS Builder dialog box that shows copyright and version information.

Table 2.7: Help menu commands.

2.3Creating the application interface

All visual design work in the CARS Builder takes place on forms. When you open the CARS Builder or create an new project, a blank form is displayed on the screen. You can use it to start building your application interfaces and dialog boxes. You design the user interface for the application by placing and arraging static and dynamic visual components on the form. Such as buttons, labels and pumps.

2.4Adding components

Components are the elements you use to build your CARS Builder applications. They include all the visible parts of an application interface as well as those that are not visible while the application is running, such as logging components.

Many visual components are provided in the development environment itself on the Component palette. You select components from the Component palette and drop them onto the form to design the user interface. Once a visual component is on the form, you can adjust its position and size. Forms and components have many features in common. You can think of a form as a component that can contain other components.

CARS Builder components are grouped functionally on the different pages of the Component palette. You can customize the Component palette by adding or deleting pages and components.

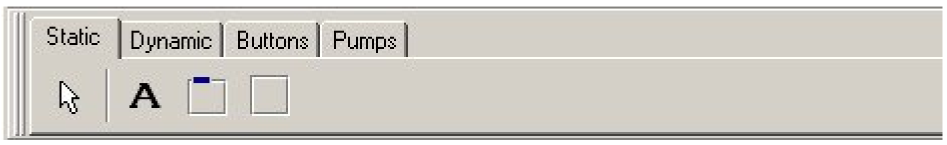
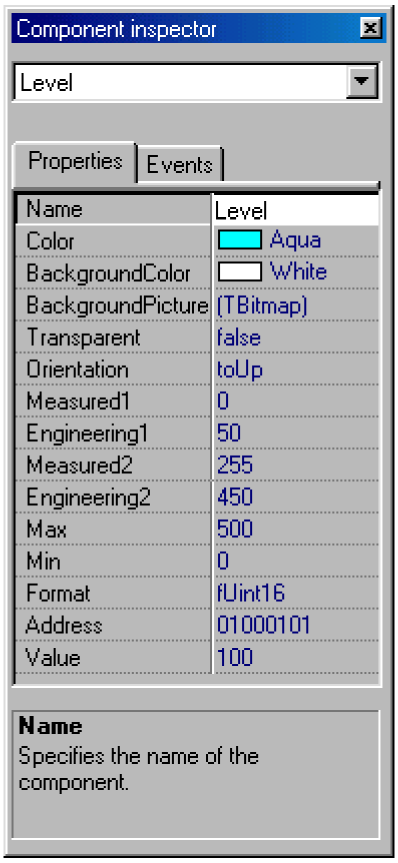


Figure 2.2: Component palette.

2.5Changing component behavior

You can customize the way a component appears and behaves in your application by using the Component inspector. When a component has the focus on the form or a form itself, its properties are displayed in the Component inspector.



Name	Level
Color	🟦 Aqua
BackgroundColor	⬜ White
BackgroundPicture	(TBitmap)
Transparent	false
Orientation	toUp
Measured1	0
Engineering1	50
Measured2	255
Engineering2	450
Max	500
Min	0
Format	fUint16
Address	01000101
Value	100

Name
Specifies the name of the component.

Figure 2.3: Component inspector.

2.6Managing project items

Using the Project explorer, you can keep track of and access the items that make up a CARS Builder application. The Project explorer shows you the forms and the visible and non visible components contained in a project. Using the explorer, you can easily navigate between them.

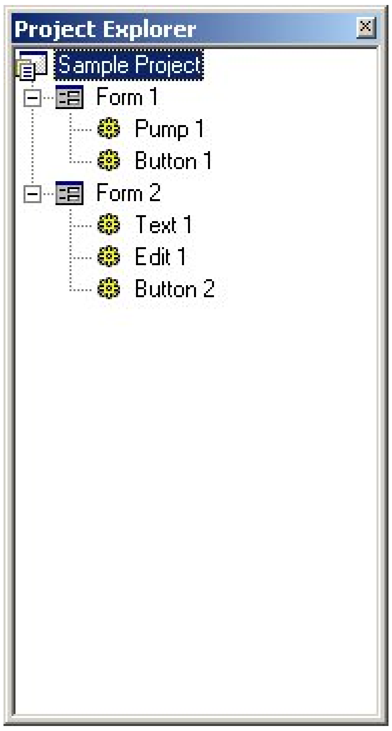


Figure 2.4: Project explorer.

2.7Storing application items

CARS Builder uses the Project, Form and Component library to store application items. The libraries lets you easily reuse the projects, forms and components that you build. Reusing items lets you build families of applications with common user interfaces and functionality. For example you can make forms that you can use in more applications and save them in the Form library.

2.8Testing applications

A CARS Builder project is saved in a CRS file. This file is used by CARS 6 to show process information and get logging information of a station. You can view and test your application by starting CARS 6.

3 Getting Started

The best way to introduce yourself to the CARS Builder is to write a quick Cars 6 application, later on it will be called it a project. The previous chapter briefly introduced the CARS Builder visual programming environment and presented an overview of the most important parts of the CARS Builder. The environment provides many tools that support your work and simplifies the development process.

This chapter briefly describes the development enviroment and touches on many of the tools that are available to you through the CARS Builder environment.

3.1Starting the CARS Builder

You can start CARS Builder the same way you start any Windows-based application. Here are some of the common ways:

- Double-click on the CARS Builder icon
- Use the Explorer or the File Manager to navigate the file system. Locate and double-click the CarsBuilder.exe file.
- Choose Run from the Windows Start menu, and specify the path of CarsBuilder.exe.

Figure 3.1 shows what CARS Builder looks like when you first start it up.

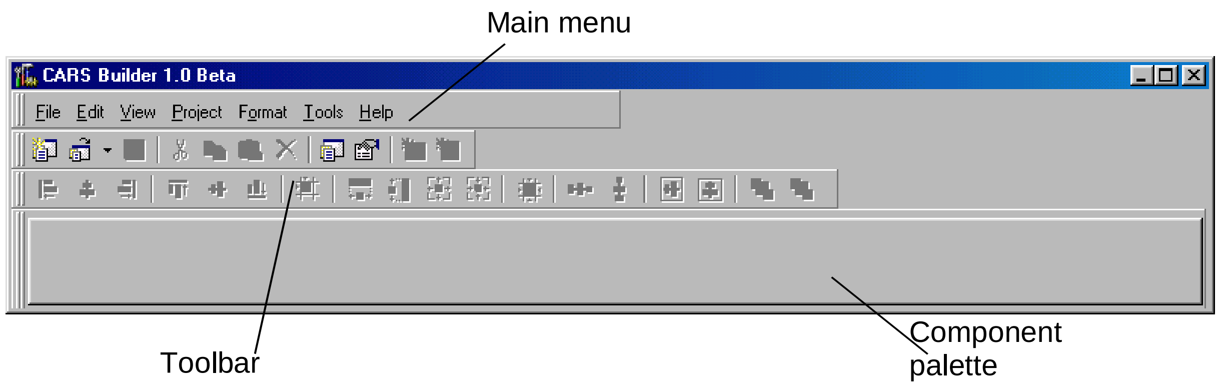


Figure 3.1: CARS Builder Development environment.

CARS Builder exist from two main parts :

- CarsBuilder.exe, the development environment.
- YPComponts.dll, the components for the CARS Builder.

You always need those two parts!

At the first time you need to import the components in CARS Builder.

3.2Import Library

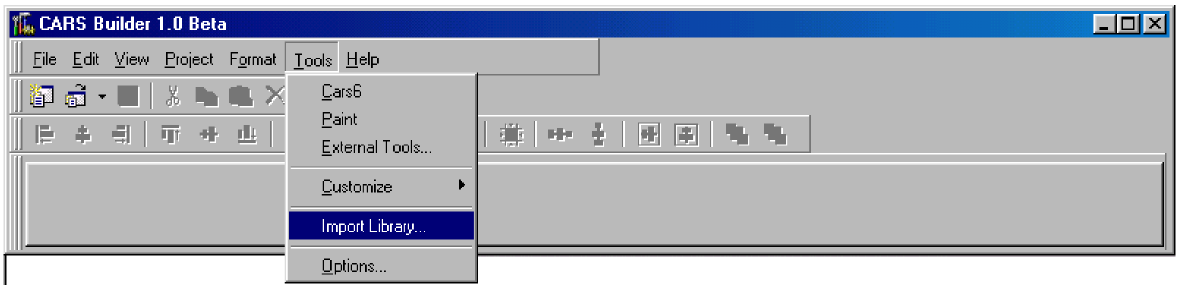


Figure 3.2: Import components in CARS Builder.

CARS Builder get the components from the YPComponents.dll library file and will place them in template directories. To change the location of the template libraries you go to Tools -> Options->Library.
After you have imported the components they are available within CARS Builder and you can start to create your project. But at first we will go to customize the component palette.

3.3Customize the component palette

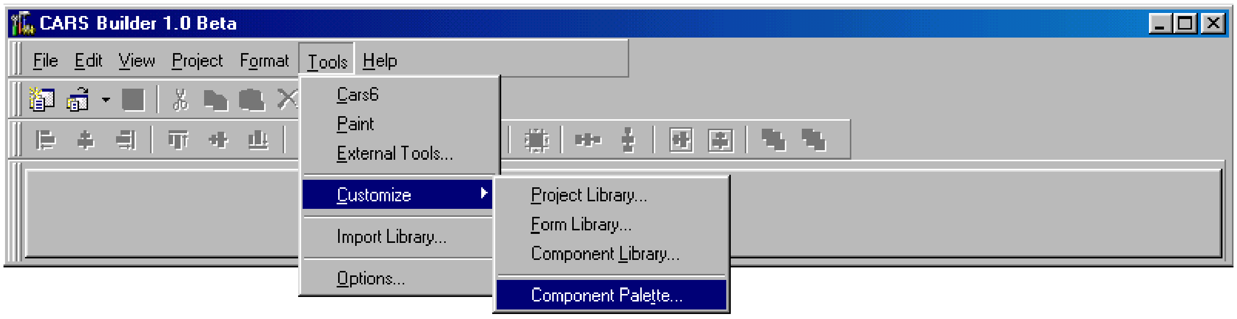


Figure 3.3: Customize the component Palette.

When you customize the component palette you are be able to get the components from the palette. It is not necessary, but it will reduce the develop time of a project because you don’t have to select the components by the menu. You also can group it the way you want. After you have selected Customize->Component Palette from the menu you will see figure 3.4.

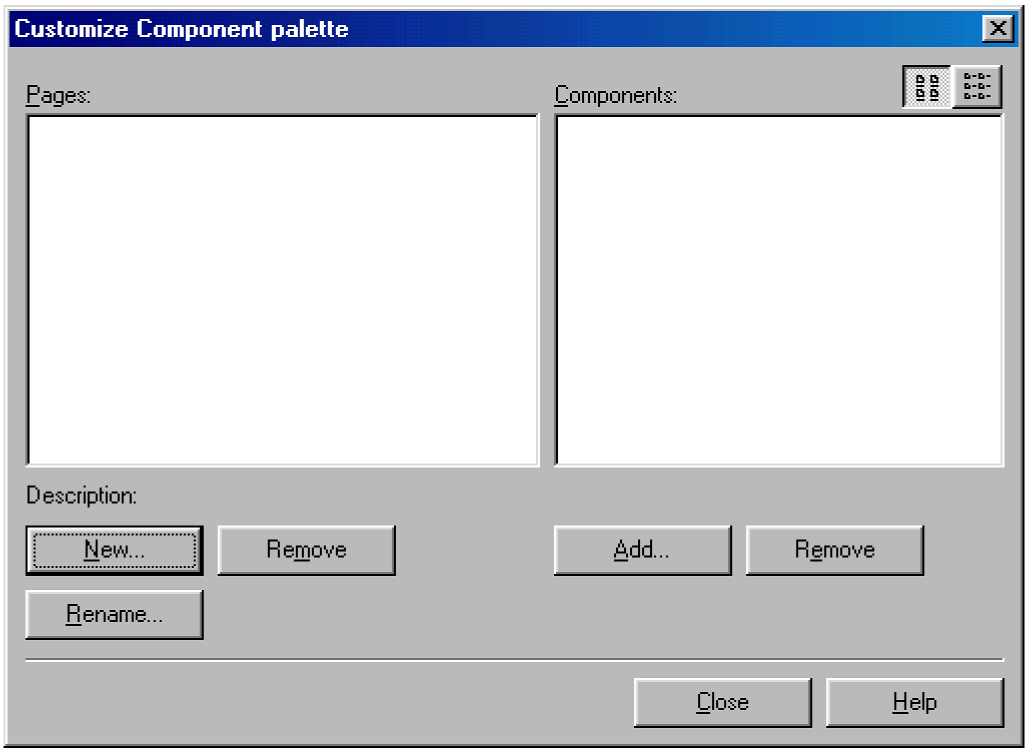
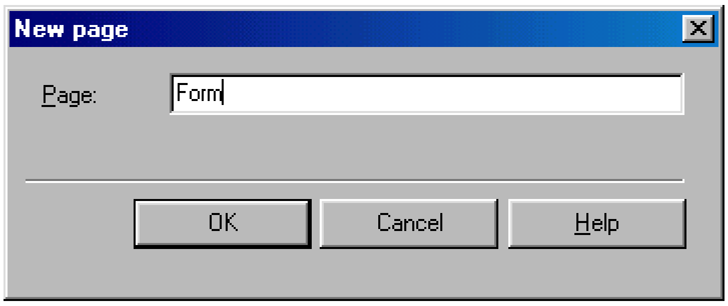


Figure 3.4: Customizing the component palette.

If you click on new you can add a new page to the component palette.



Page: Form

Figure 3.5: Add the page Form to the component palette.

Now you can add some components to this page. There for you have to click on the Add button and you will get figure 3.6.

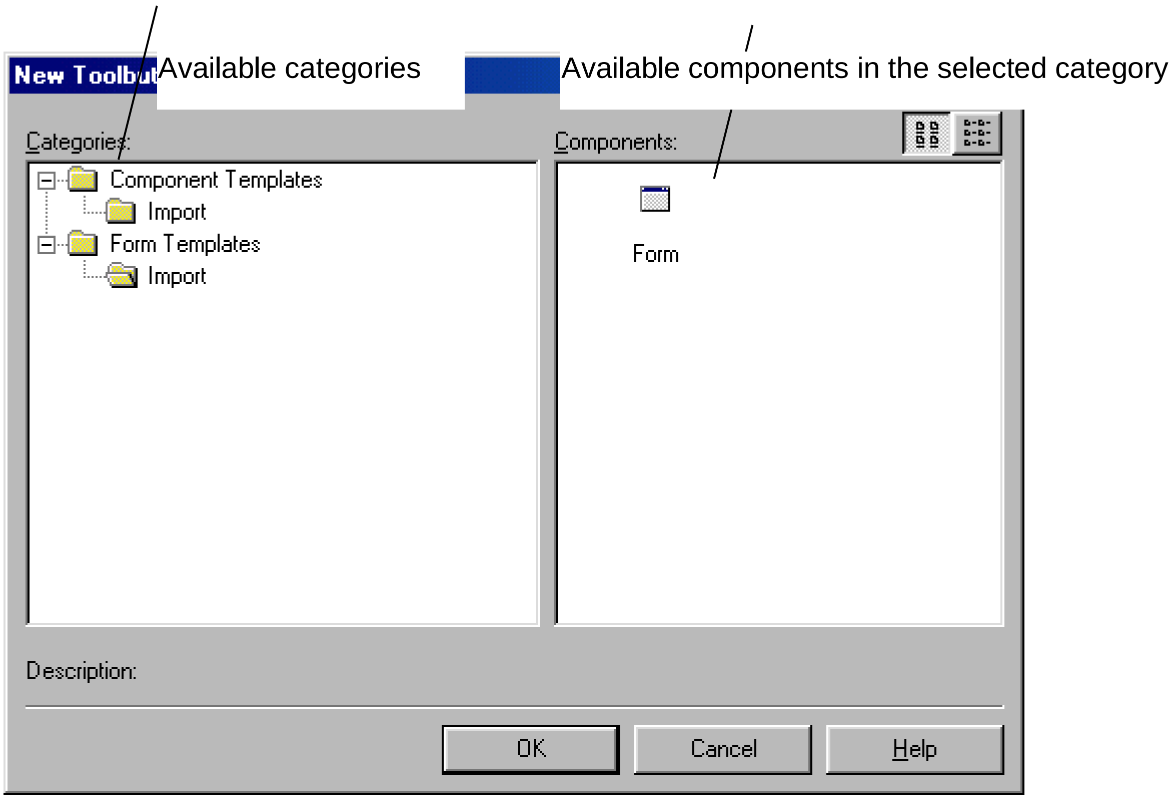


Figure 3.6: Add a new toolbutton to the page Form in the component palette.

If you import the library, CARS Builder will create the import categories and the components which are in these categories.
Select in the treeview Categories the category Form Templates->Import. You will see in the Components treeview a component named Form. Select it and click on the OK button. Now you have created a page, named Form, on the component palette with one toolbutton Form. You can create more pages and dropping some components on it so it can look like figure 3.7

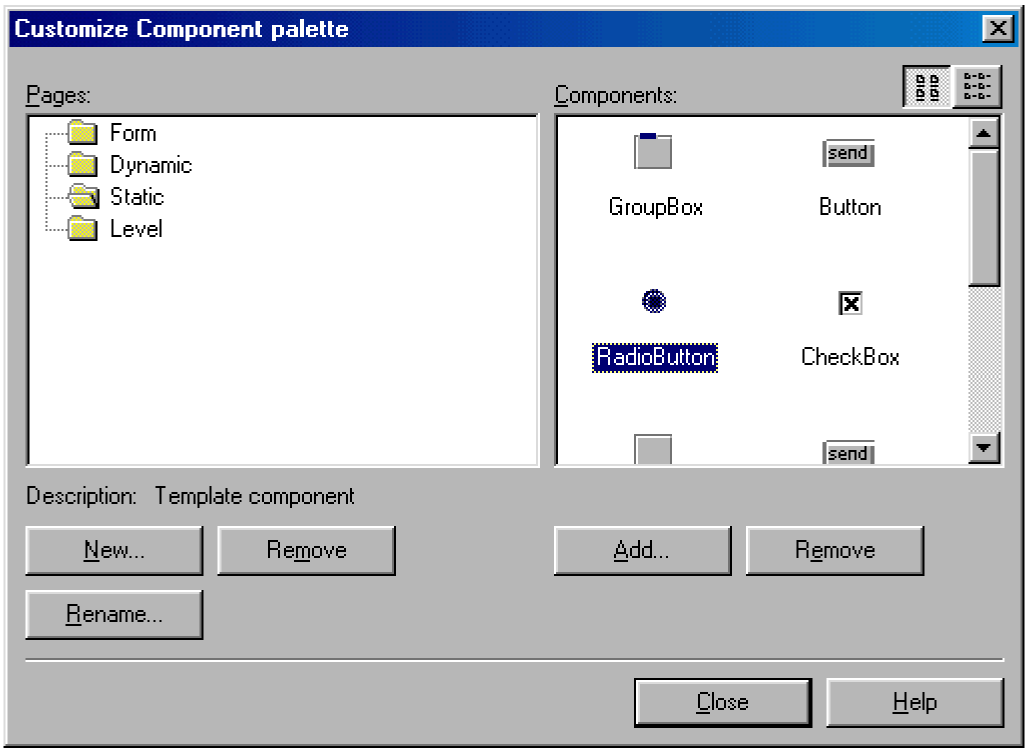


Figure 3.7: Component palette customized.

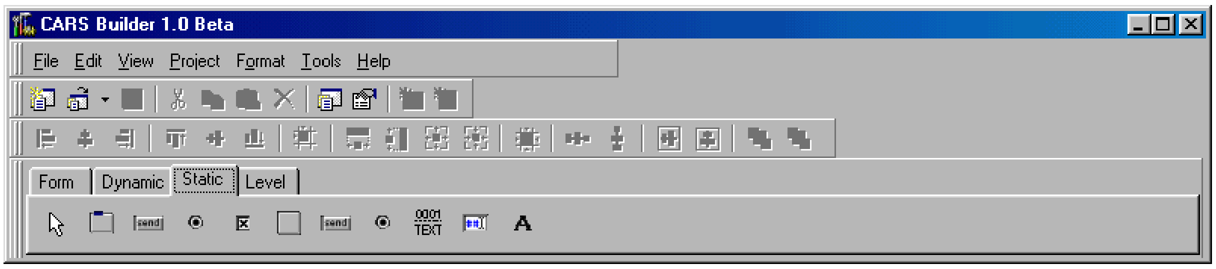


Figure 3.8: How it looks on the component palette.

You are now be able to get your components from the palette. So we will now start to create a new project.

3.4Create a new project

To create a new project you always need a template project. In our case we don’t yet have any existing projects so we need the Blank Project which is imported from the YPComponents.dll library file.

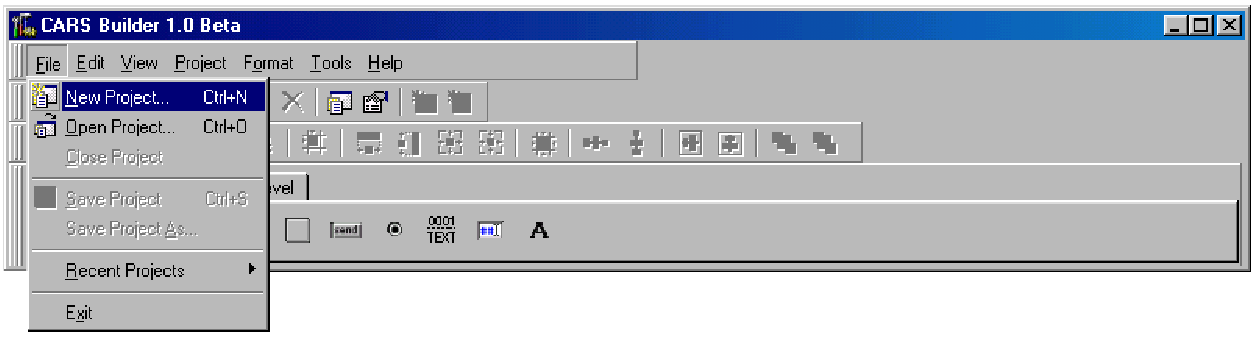
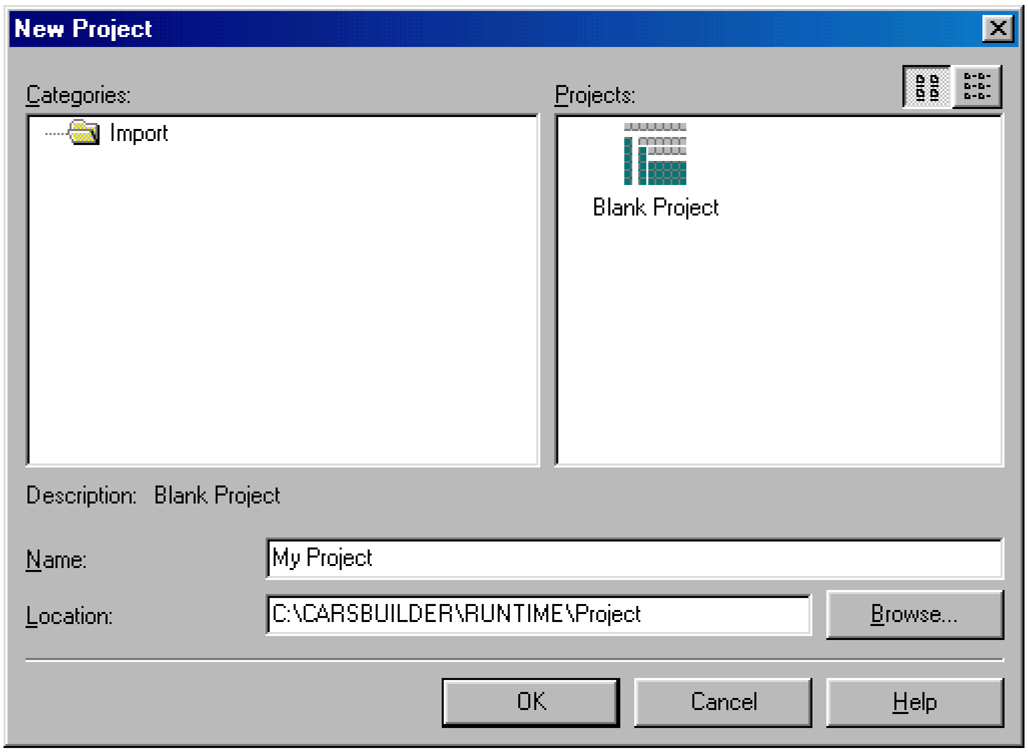


Figure 3.9: Create a new Project.



Description: Blank Project

Name: My Project

Location: C:\CARSBUILDER\RUNTIME\Project

Figure 3.10: Creating a new project named My Project.

First select a category then a template project, give it a name and select the location where you want the project to be stored. Click on OK and you will see something like figure 3.11.

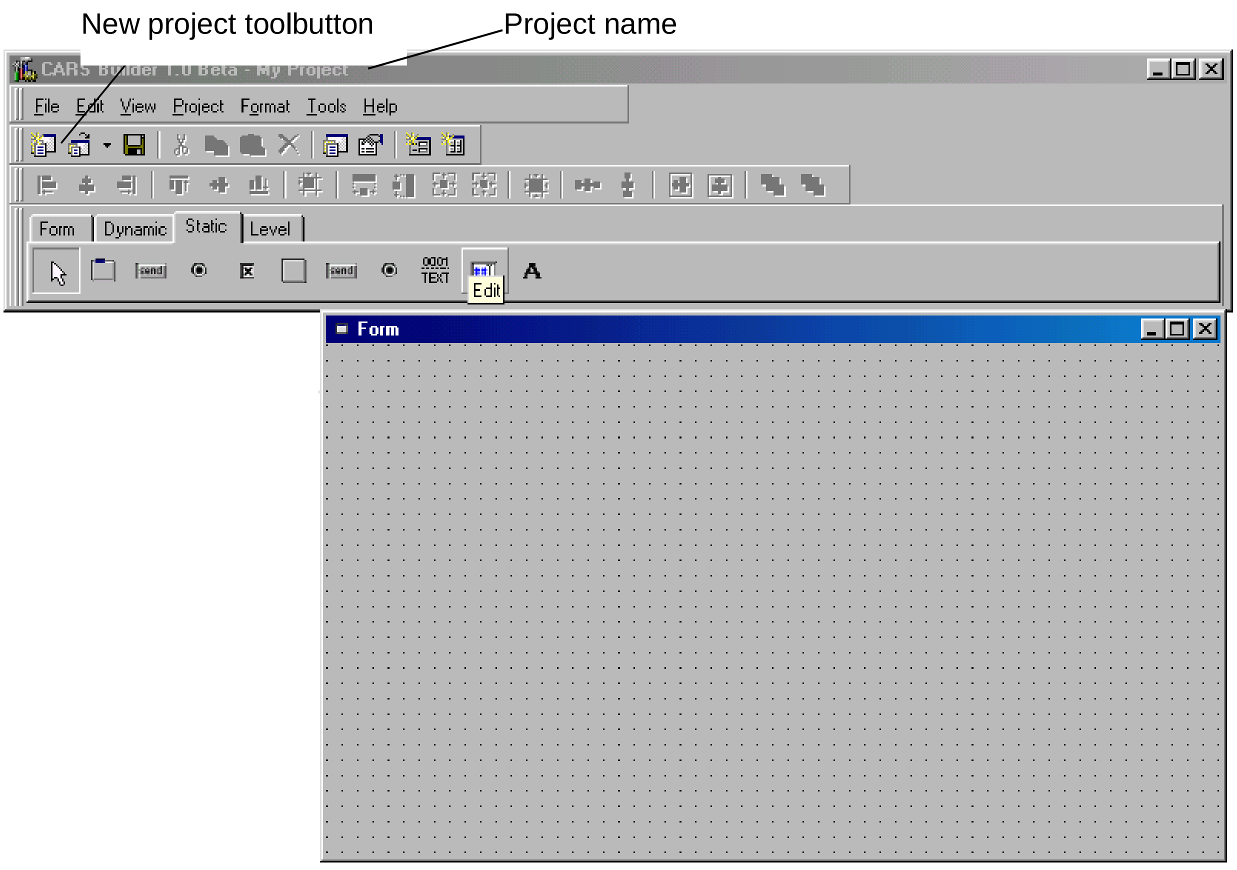

New project toolbutton
Project name
CARS Builder 1.0 Beta - My Project
File Edit View Project Format Tools Help
Form Dynamic Static Level
Edit
Form

Figure 3.11: My Project

When you create an new project CARS Builder displays a form, named Form by default. In CARS Builder, you can now design the user interface of your project, but first we are going to set the project options.

3.5Setting the project options

Go to Project->Options.

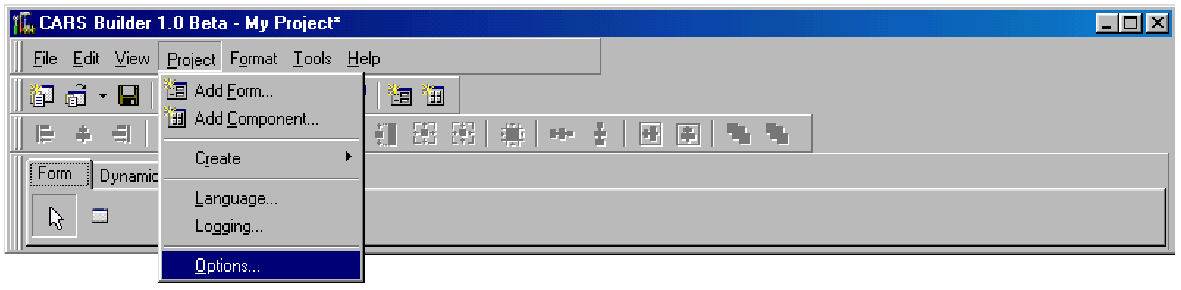
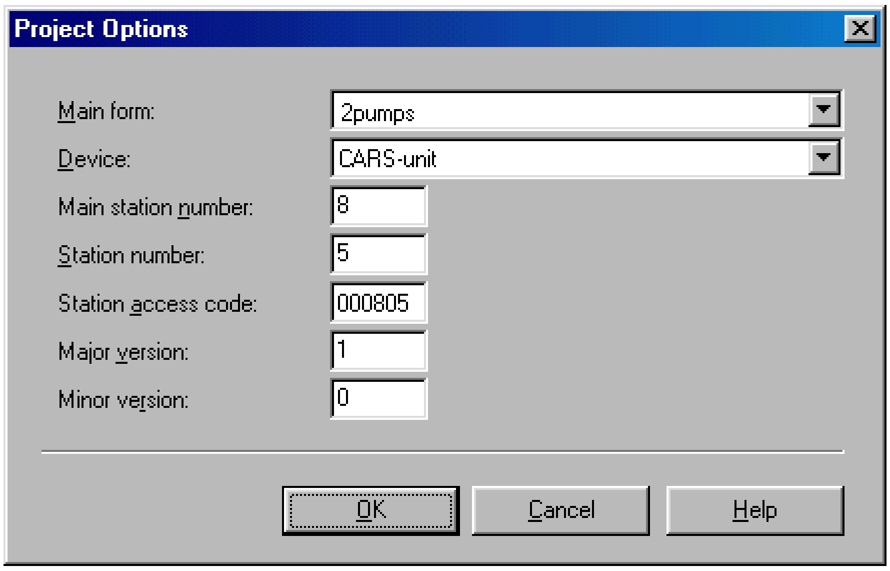


Figure 3.12: Setting the project options

De project options dialog box will be visible.



Main form:	2pumps
Device:	CARS-unit
Main station number:	8
Station number:	5
Station access code:	000805
Major version:	1
Minor version:	0

Figure 3.13: Project Options

The option main form lists the forms currently available in the project. We now have one form called Form so this is also our main form. The main form is the form users see first when a connection with a station is established, the other forms are, at the beginning, not visible.
Use the option device to set the device located at the station. If the device is “None”, which is default, CARS Builder will not check for a valid address, when you modify the address property of a component.
The other options will be explained later on in this manual.
Click on the OK button to keep the changes.

3.6Placing a component

You place components on forms to create user interfaces. A component can be selected from the component palette, the menu or the toolbar. When you select a component from the component palette you can drop it on a control where you want. This control can be a Form, GroupBox, Panel or Level and will be the parent of the component.

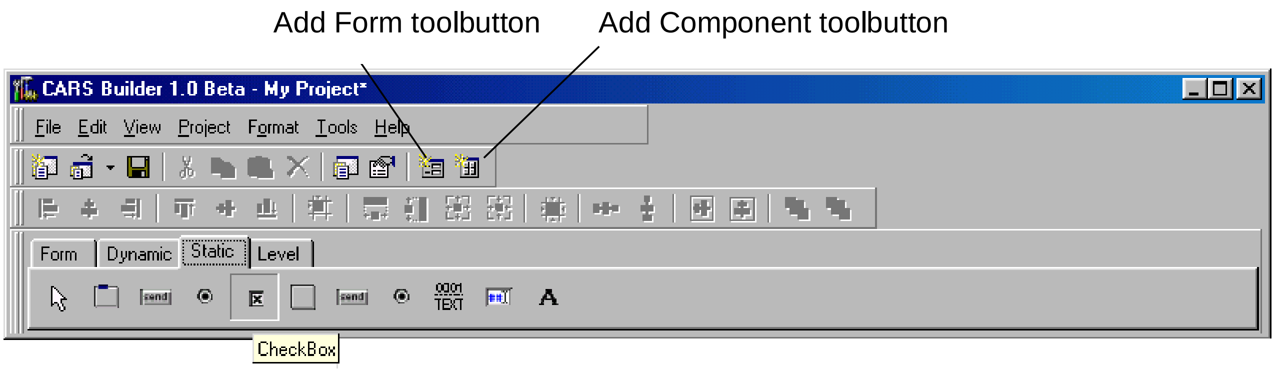

Add Form toolbutton
Add Component toolbutton
CARS Builder 1.0 Beta - My Project*
File Edit View Project Format Tools Help
Form Dynamic Static Level
CheckBox
Selector button
CheckBox button selected

Figure 3.14: Select a component from the component palette.

By double-click, on the component palette, the component will be placed in the centre of it’s parent control. When you select a component from the toolbar or menu the component will also be placed in the centre of it’s parent control. If you want that a particular component is the parent, and this component is a Form, GroupBox, Panel or Level, you have to select this component and then drop the new component on it.
Select the CheckBox button on the component palette, the CheckBox button will appear lowered, and drop it on the form Form. The CheckBox is now placed on Form. Now the Selector button will be lowered and the CheckBox button is normal, so you can select the CheckBox on Form and modify it. With the Selector button selected you can select components on a form. With other buttons selected you will be able to place components.

Moving and resizing a single component

To move the component with the mouse:
1. Select the component on the form.
2. Hold the left mouse button down and drag the component to a new location.

To resize a component with the mouse:
1. Select the component.
2. Move the mouse cursor to one of the black nibs at the component edge or corner. When it turns into a double arrow, hold the left mouse button down and drag to the desired size.

To move a component with the keyboard:

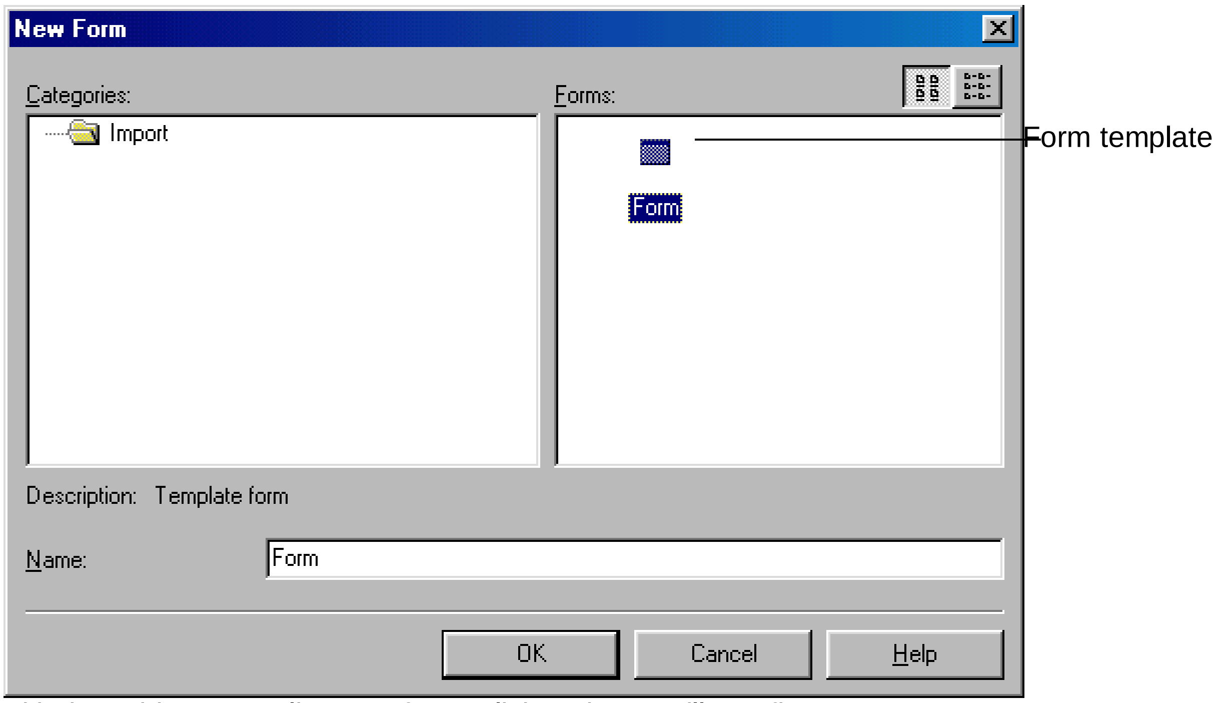
1. Press Tab until the component you want is selected.
2. Press the Ctrl and Up arrow key to move the component up; Press the Ctrl and Down arrow key to move it down. Press the Ctrl and Right arrow key to move the component to the right; Press the Ctrl and Left arrow key to move it to the left. If you want to move the component to the next grid: Hold Shift and Ctrl key down and with a arrow key you can snap to the next grid.

To resize a component with the keyboard:

1. Press Tab until the component you want is selected.
2. Press the Shift and Up arrow key to decrease the component height; Press the Shift and Down arrow key to increase it. Press the Shift and Right arrow key to increase the component width; Press the Shift and Left arrow key to decrease it.

3.7Placing a Form

A Form can be placed on the same way a component is placed. Now we will place a Form



Description: Template form

Name: Form

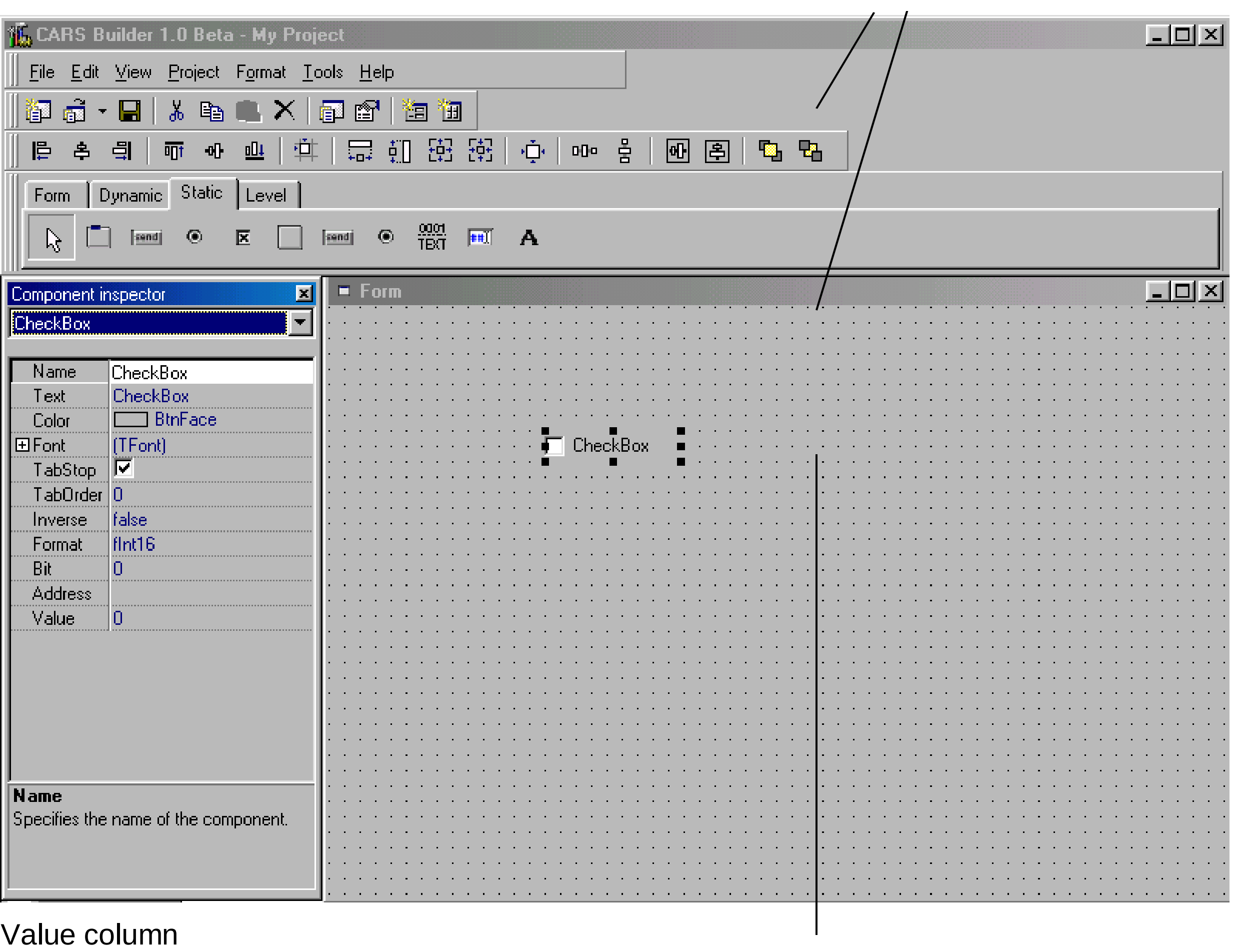
with the Add Form toolbutton. If You click on it you will see figure 3.15.

Figure 3.15: Selecting a Form

Now you select a Form template and give it a name. If the name already exist CARS Builder will change the name. The name Form already exist so CARS Builder changed it to Form1. Now we have a project with two Forms. On the form named Form we have a CheckBox and the form Form1 is still blank.

3.8Setting property values

You can easily customize the way a component appears and behaves in your project by using the Component Inspector. You can get the Component inspector visible by the menu : View -> Component Inspector, or toolbar or by short cut F11.



Name	CheckBox
Text	CheckBox
Color	BtnFace
⊞Font	(TFont)
TabStop	✓
TabOrder	0
Inverse	false
Format	fInt16
Bit	0
Address	
Value	0

Name
Specifies the name of the component.

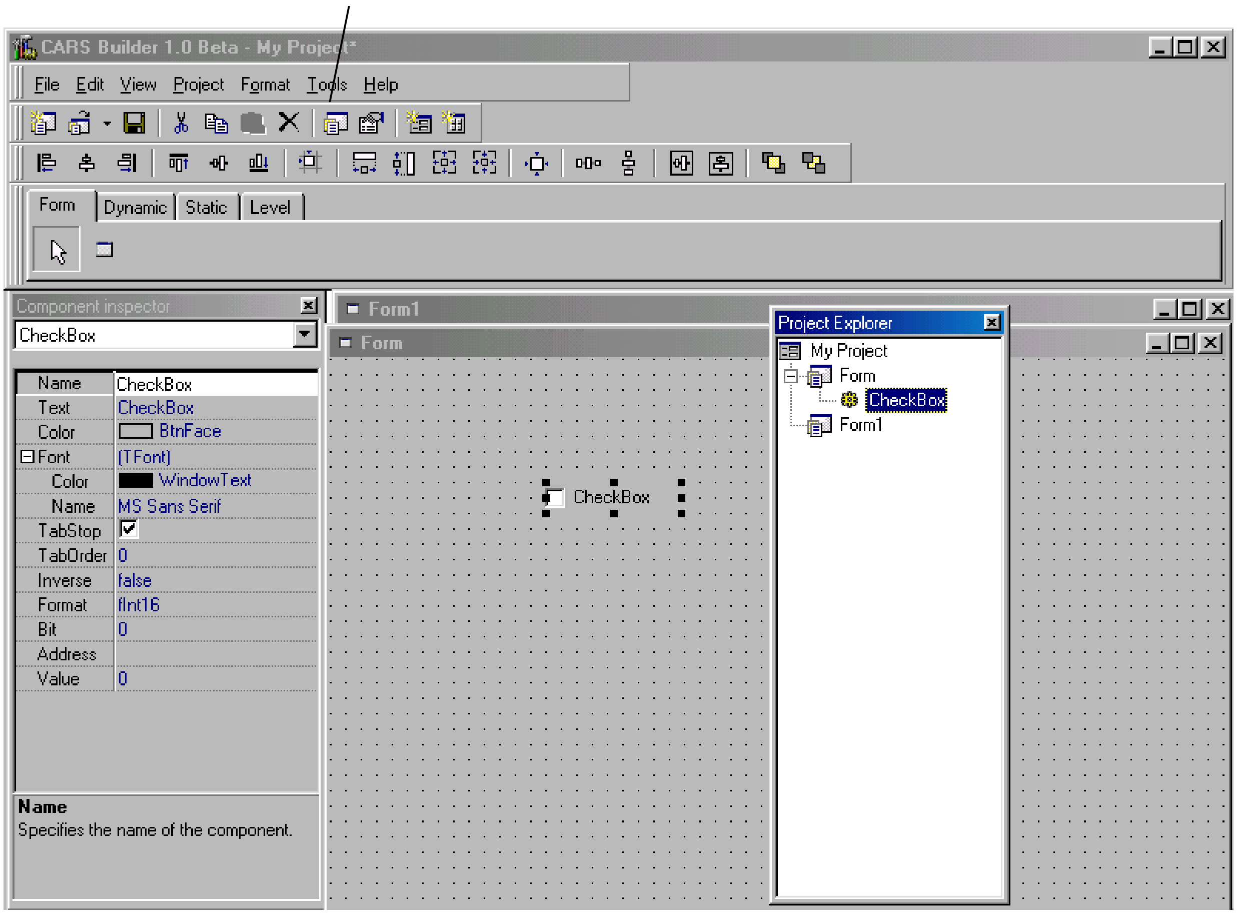
Figure 3.16: My Project with a checkbox on a form and the Component inspector visible.

When the Form has the focus, the component inspector will show the properties of the form. You can change the values of the properties, Value column, to change the behaviour of the component.
With the component selector you can select components, which are on the selected form, and see his properties.

3.9Exploring the project

With the Project Explorer you can easily navigate through the project.

Project explorer toolbutton



CheckBox

Name	CheckBox
Text	CheckBox
Color	BtnFace
⊟Font	(TFont)
Color	WindowText
Name	MS Sans Serif
TabStop	✓
TabOrder	0
Inverse	false
Format	fInt16
Bit	0
Address	
Value	0

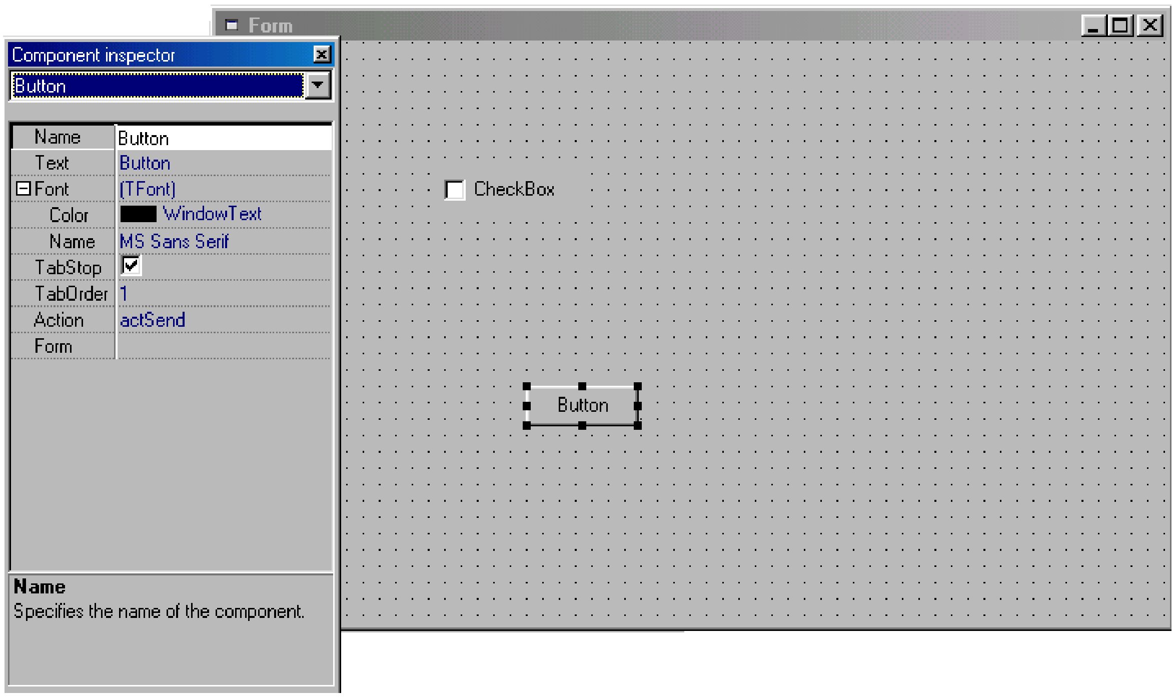
Name
Specifies the name of the component.

Figure 3.17: Using the Project Explorer

The project explorer will be visible by selecting it from the toolbar, menu or by short cut ctrl+alt+p. In the project explorer you can see all the components and forms that are within the project. You can also select them in the project explorer.

3.10Add buttons to the project

To make your project more user-friendly you need to add some buttons so the user can switch to a form and close a form when the project is loaded in CARS. At first we select the Button component and place it on the form named Form.

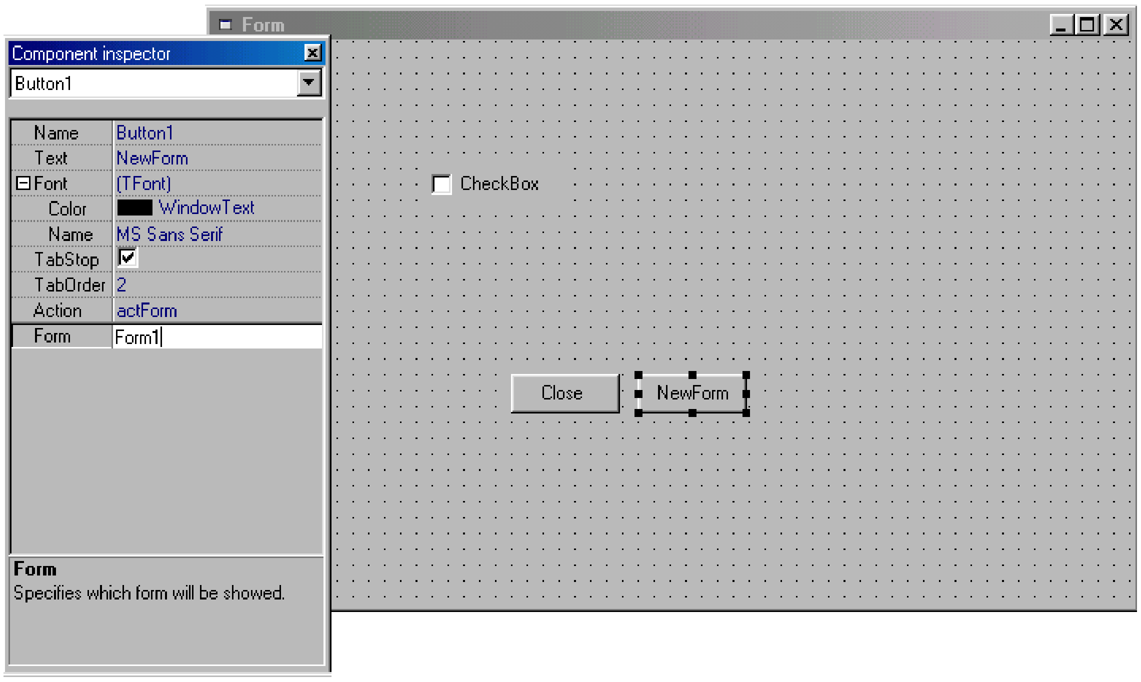


Name	Button
Text	Button
⊟Font	(TFont)
Color	■ WindowText
Name	MS Sans Serif
TabStop	✓
TabOrder	1
Action	actSend
Form	

Name
Specifies the name of the component.

Figure 3.18: A button component on Form

Change the property Text in Close and Action in actClose. Now you have created a close button. If the user will click on it in CARS, the form will be closed.
Add another button to Form. Change the properties Text to NewForm and Action to actForm. Now you need to specify the name of the form that will be showed if you click on it in CARS. Change the Form property to Form1, so Form1 will be showed if you click on this button. Form should now look like figure 3.19.



Name	Button1
Text	NewForm
⊟Font	(TFont)
Color	■ WindowText
Name	MS Sans Serif
TabStop	✓
TabOrder	2
Action	actForm
Form	Form1

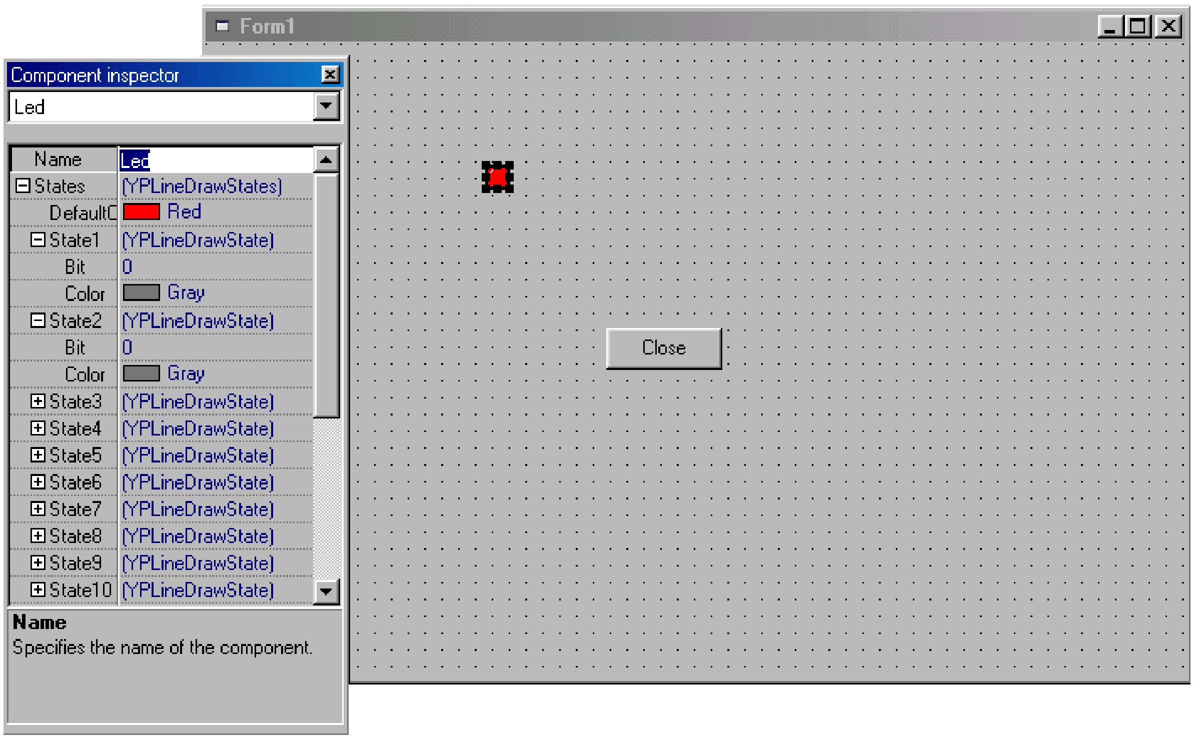
Form
Specifies which form will be showed.

Figure 3.19: Form with a CheckBox and two buttons.

In the project options is set that Form is the MainForm. So if you close in CARS the form Form, all the forms belonging to the project will be closed.
You can also add a close button on Form1 so the user can easily close this form.
Now we have made a little project where you can switch between two forms.

3.11Create a template

First add a Led component to Form1.



Name	Led
⊟States	(YPLineDrawStates)
DefaultC	🟥 Red
⊟State1	(YPLineDrawState)
Bit	0
Color	⬛ Gray
⊟State2	(YPLineDrawState)
Bit	0
Color	⬛ Gray
⊞State3	(YPLineDrawState)
⊞State4	(YPLineDrawState)
⊞State5	(YPLineDrawState)
⊞State6	(YPLineDrawState)
⊞State7	(YPLineDrawState)
⊞State8	(YPLineDrawState)
⊞State9	(YPLineDrawState)
⊞State10	(YPLineDrawState)

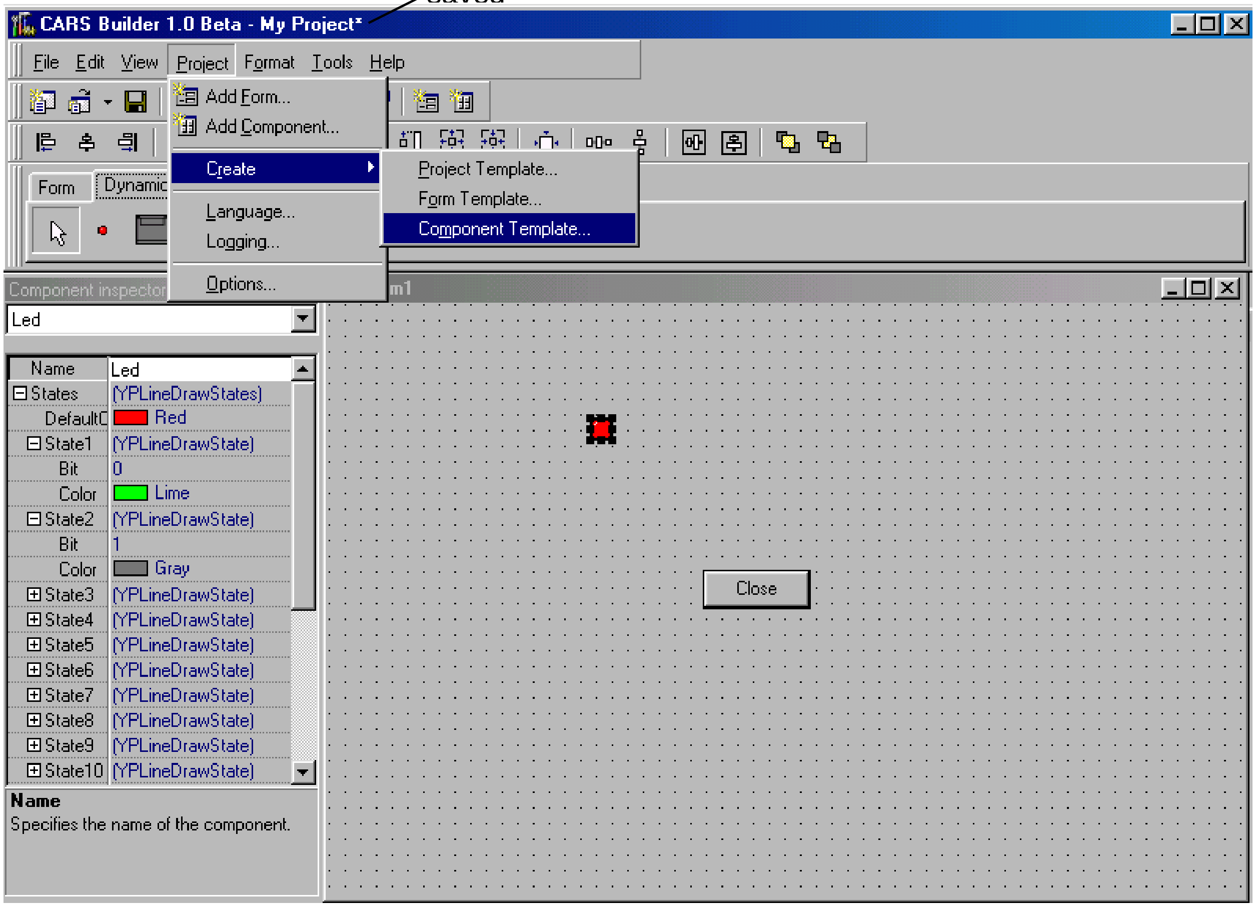
Name
Specifies the name of the component.

Figure 3.20: A Led component placed on Form1.

Then change the properties States->State1->Color to Green, States->State2->Bit to 1. When the property Value is 0 the led will be red if Value is 1 the led will be green and by Value is 2 the led will be gray. You can check states by changing the Value property.
Now we have created a led component with a default color, red, and two states : green on bit 0 and gray on bit 1. If you want to use this component frequently you can make a template component of it.

To do this, select the component and then Create->Component Template and you will see figure 3.22.

The * appears if the project is changed and not yet saved

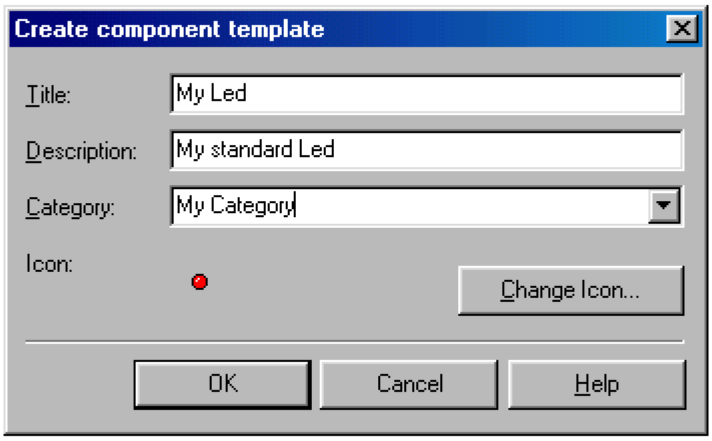


Led

Name	Led
⊟ States	(YPLineDrawStates)
DefaultC	🟥 Red
⊟ State1	(YPLineDrawState)
Bit	0
Color	🟩 Lime
⊟ State2	(YPLineDrawState)
Bit	1
Color	▨ Gray
⊞ State3	(YPLineDrawState)
⊞ State4	(YPLineDrawState)
⊞ State5	(YPLineDrawState)
⊞ State6	(YPLineDrawState)
⊞ State7	(YPLineDrawState)
⊞ State8	(YPLineDrawState)
⊞ State9	(YPLineDrawState)
⊞ State10	(YPLineDrawState)

Name
Specifies the name of the component.

Figure 3.21: Select a component and creating a template component of it.



Title:	My Led
Description:	My standard Led
Category:	My Category
Icon:	●

Figure 3.22: Creating a component template

Now you can give the component a new name, for example My Led, and it will be saved as My Led.cct in the directory you specify with category. This new component can now also be added to the component palette.
The same way you create a component template you can also create a form- or project template.

3.12Save the project

The project is changed so you have to save it, if you want to keep the changes. You can do it by the menu or toolbutton.

Save toolbutton

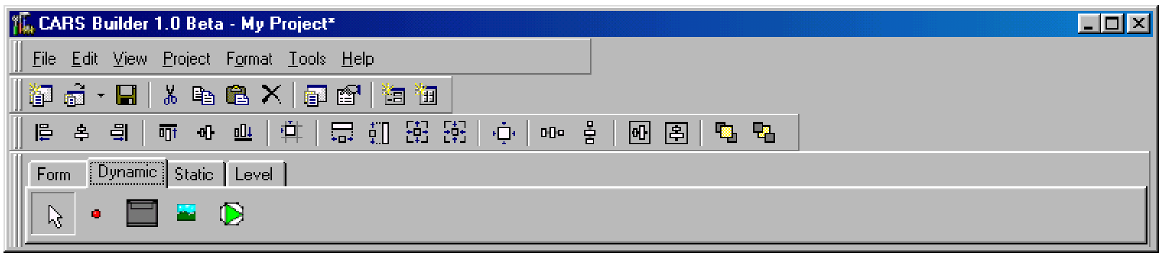
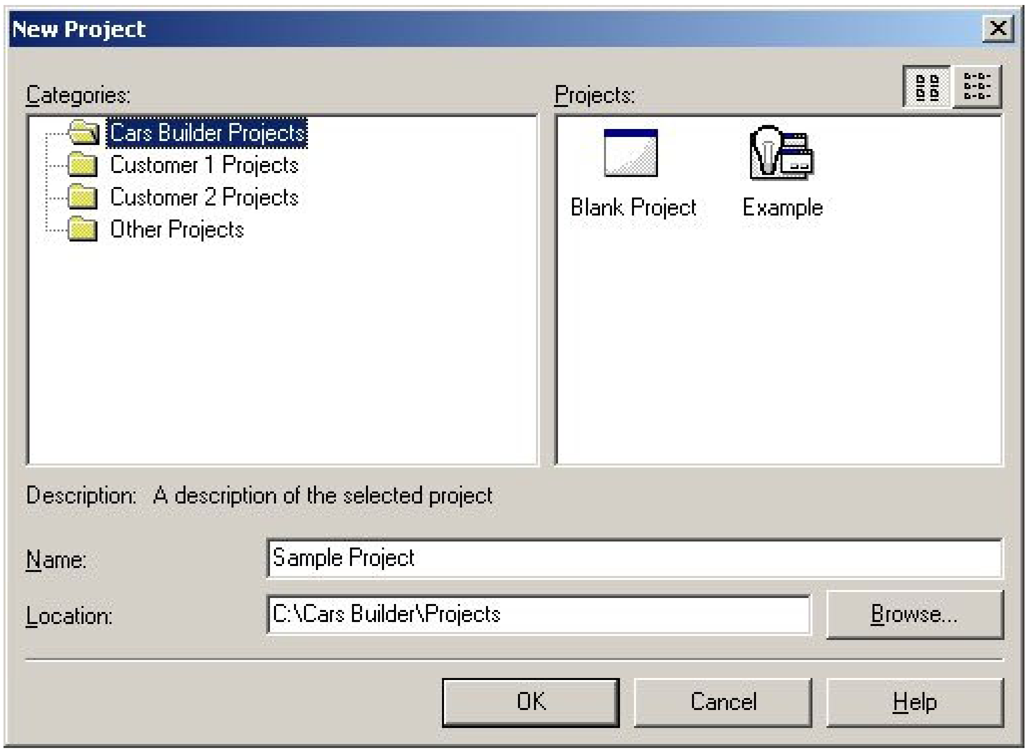

CARS Builder 1.0 Beta - My Project*
File Edit View Project Format Tools Help
Form Dynamic Static Level

Figure 3.23: Saving the project.

4 Creating and managing projects

4.1Creating a new project

Use the New Project dialog box to create a new project. If a project is currently open, you are prompted to save your changes and the currently open project is closed before you create the new project. You can access this dialog box by choosing New Project from the File menu.



Description: A description of the selected project

Name: Sample Project

Location: C:\Cars Builder\Projects

Figure 4.1: New Project dialog box.

Categories
Displays the categories of projects available.

Projects
You can select a Project in the right pane. A brief description of the selected project appears beneath Categories. A project template creates the initial forms, components and property settings appropriate for the selected project.

Large Icons
Displays projects as large icons with a label below it.

List
Displays projects as small icons with a label to the right of it.

Name
Enter here the name of the project you want to create. The name is displayed in the title bar of CARS Builder.

Location
Enter here the location where you want to create your project. By default, new projects are created in the folder Projects of the CARS Builder folder.

Browse
Displays the Project Location dialog box, which allows you to navigate to a new folder to save the project in.

4.2Opening a project

Use the Open Project dialog box to open an existing project. If a project is currently open, you are prompted to save your changes and the currently open project is closed before you open another project. The name of the project is displayed in the title bar of CARS Builder. You can access this dialog box by choosing Open Project from the File menu.

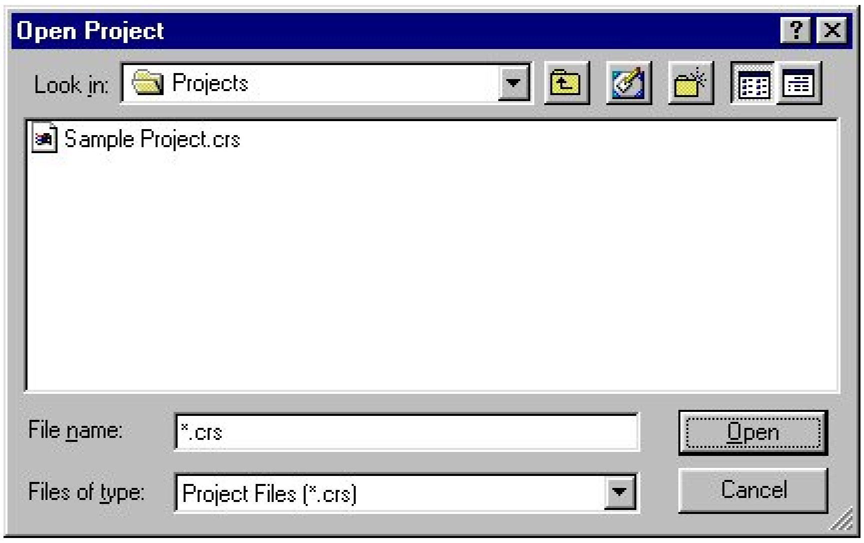


Figure 4.2: Open Project dialog box.

Look in
Locate the existing project folder from this list. Selecting a folder from this list displays the contents of the folder in the primary pane. Default the last folder used will be shown.

File name
Use this option to filter the files and folders that are displayed. Enter a full or partial file name on which to filter. You can use the asterisk (*) as a wildcard.

Files of type
Use this option to filter the files displayed based on file extension. Default files with the file extension CRS are shown.

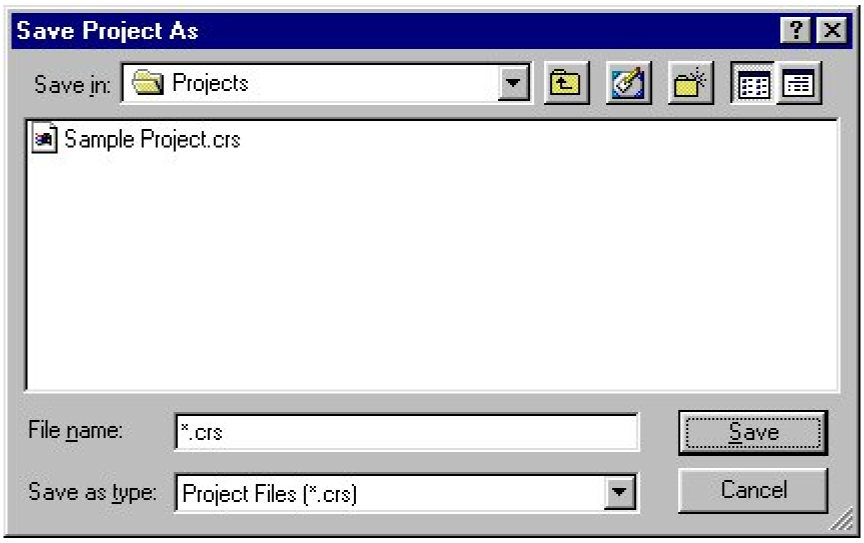
4.3Closing a project

You can close the currently open project by choosing Close Project from the File menu. Before closing the project, CARS Builder prompts you to save any changes.

4.4Saving a project

Save project
Choose Save from the File menu to store changes made to all items included in the open project using the current name for the project.

Save project as
Use the Save Project As dialog box to change the project file name or to save the project in a new location. If the file name already exists, CARS Builder asks if you want to replace the existing file. You can access this dialog box by choosing Save Project As from the File menu.


Save Project As

Save in: Projects

Sample Project.crs

File name: *.crs

Save as type: Project Files (*.crs)

Figure 4.3: Save Project As dialog box.

Save in
Locate an existing folder from this drop-down menu. Selecting a folder from this list displays the contents of the folder in the primary pane

File Name
Enter a name for the project file you are saving.

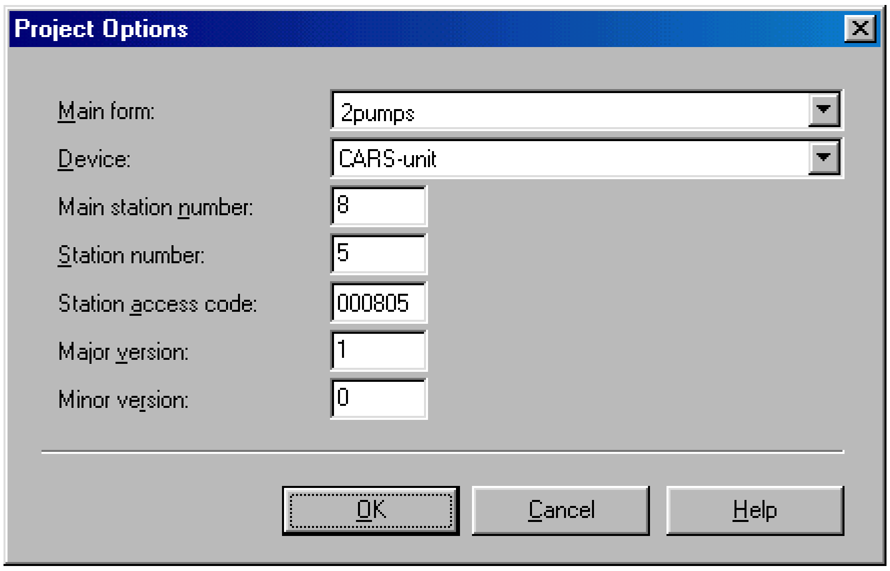
Save as type
Choose a file extension; the default is CRS. All files in the current directory of the selected type appear in the primary pane. Note that saving a project file with a different extension can not be used by CARS 6.

4.5Recently used projects

Choose Recent Projects from the File menu to reopen a recently closed project. When you close or save a project, it is added to the Recent Projects list.

4.6Setting project options

Use the Project Options dialog box to set project options. Project settings affect the current project only. You can access this dialog box by choosing Options from the Project menu.



Main form:	2pumps
Device:	CARS-unit
Main station number:	8
Station number:	5
Station access code:	000805
Major version:	1
Minor version:	0

Figure 4.4: Project Options dialog box.

Main form
Lists the forms currently available in the project. Select the form users see first when a connection with a station is established.

Device
Use this option to set the device located at the station.

Main station number
With this option you can specify which main station belongs to the station. Only the main station which is specified by this option may connect to the station.

Station number
This option specifies the number of the station in the CARS database.

Station access code
This access code will be send by the main station when connecting to a station. The device on location checks the acces code and determines if the main stationis allowed to connect.

Major version
CARS Builder sets this property to one when the project is first created. You can use this field to track major changes to the project.

Minor version
CARS Builder sets this property to zero when the project is first created. You can use this field in conjunction with the major version to track your project’s development.

4.7Using the Project explorer

The Project explorer provides you with an organized view of your project. The tree view shows a list of the forms and components that exist in your project. You can expand or collapse the nodes on the tree. The tree view lets you quickly navigate to a form or a component. To view a specific form or component, click on the item in the tree view. CARS Builder displays the selected form or component. To access the Project explorer select Project Explorer on the View menu.

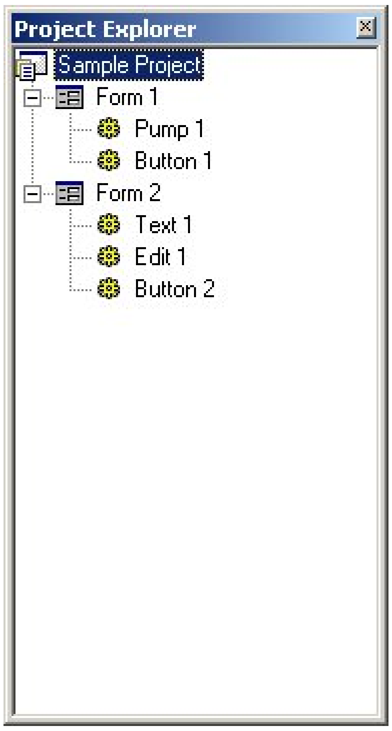


Figure 4.5: Project explorer.

4.8Creating project templates

CARS Builder is designed with the principle of reusable components in mind. This encompasses larger elements such as forms and even entire projects. You can add an entire project to the Project Library as a template for future projects. If you have a number of similar applications, you can base them all on a single, standard project.

You can add your project to the Project Library by creating a project template. Select Create Project Template form the Project menu to add the currently open project to the Project Library. The Create Project Template dialog box appears.

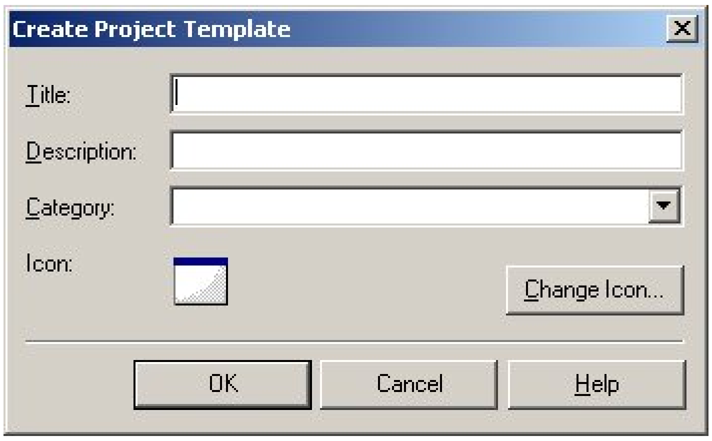


Figure 4.6: Create Project Template dialog box.

Title
Specify a template name for the currently open project.

Description
Enter a brief description for the project template.

Category
Choose a category in which the project should appear in the New Project dialog box. If you specify a category that does not exist, a new category is created.

Icon
To specify an icon for the project template, choose the Change Icon button. Navigate the Open Icon dialog box to find a directory of images, and choose a new bitmap for the icon. The bitmap you choose must be no larger than 32 pixels by 32 pixels.

The next time you choose New Project from the File menu and click the Category you selected above, your project template appears in the Projects list, with the icon and title you chose.

4.9Customizing the Project library

Use the Customize Project Library dialog box to customize the Project library. You can add, rename, or remove categories and project templates. You can access this dialog box by choosing Customize Project Library from the Tools menu.

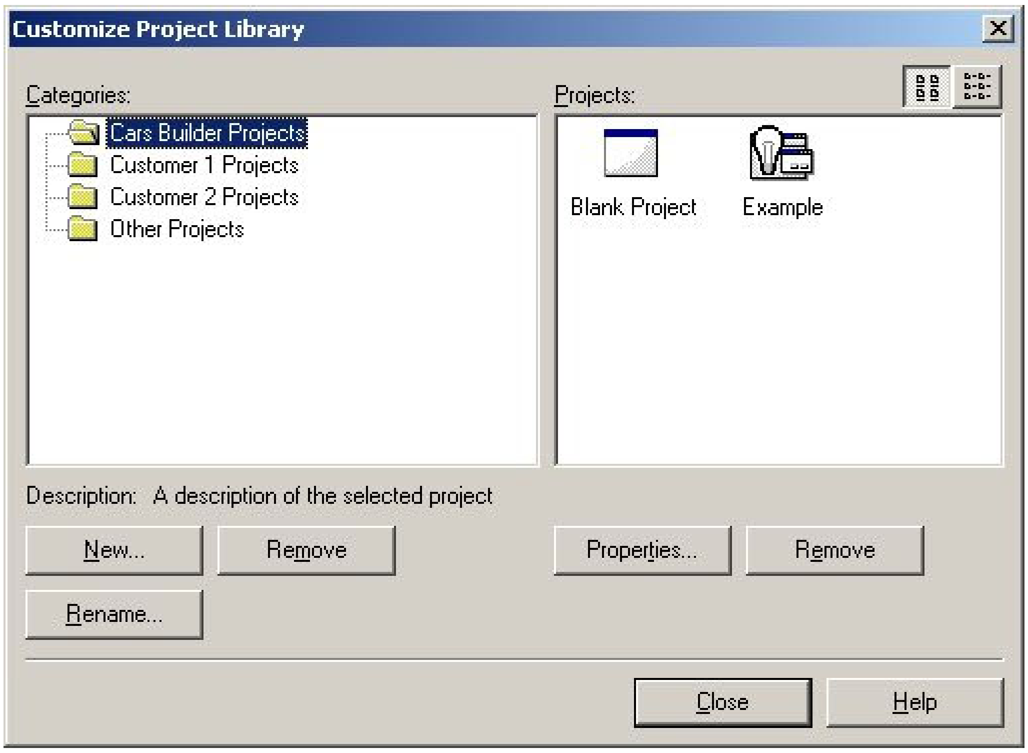


Figure 4.7: Customize Project Library dialog box.

Categories
Lists the categories in the Project library. You can rearrange these categories or view and rearrange their project templates in the Projects list.
Projects
Lists the project templates of the currently selected category in the Categories list.

Large Icons
Displays project templates as large icons with a title below it.

List
Displays project templates as small icons with a title to the right of it.

Use the following buttons when an item is selected in the Categories list.

New
Click New to display the Add New Category dialog box, where you can create a category. Once you have created a new category, you can move project templates from other categories into it or add new project templates to it.

Remove
Removes the selected category from the list. Before you can delete a category, it must be empty of project templates.

Rename
Click Rename to display the Rename Category dialog box, where you can rename the selected category.

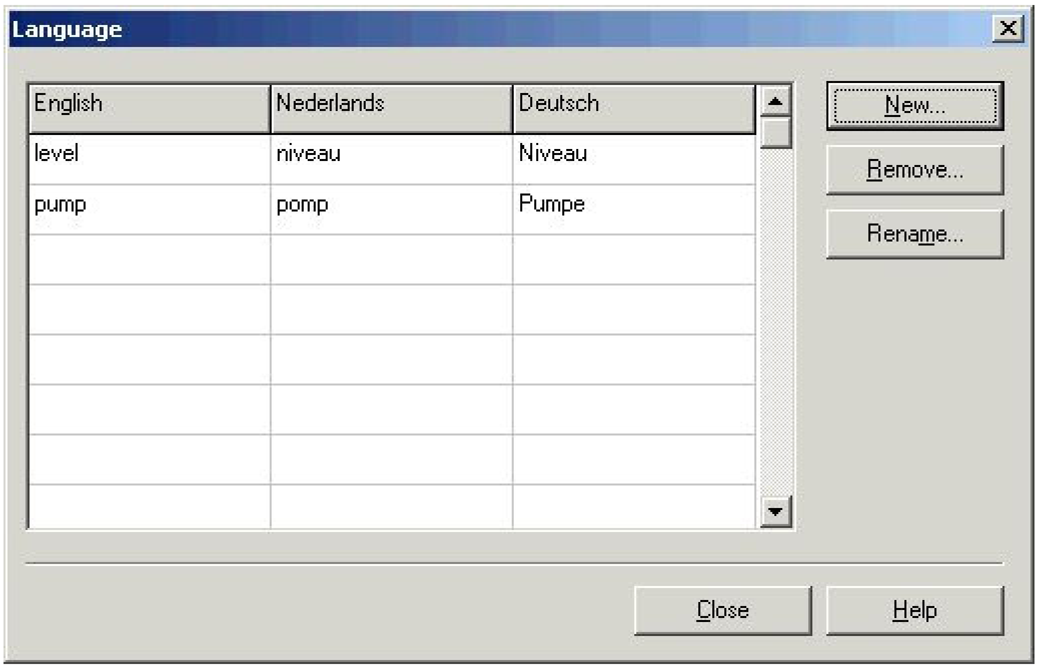
Use the following buttons when an item is selected in the Projects list.

Properties
Click Properties to display the Project Template Properties dialog box, where you can change the title, description, category and icon of the project template.

Remove
Removes the selected project template from the Project Library.

4.10Multi language support

CARS Builder supports multi language applications. You can create additional lanuages for an application by choosing Language from the Project menu. It is recommended that you first create your complete application and then create additional languages for the application. Languages are saved in seperated text files. You can edit the text in CARS Builder or by another application, like Notepad or Excel.



English	Nederlands	Deutsch
level	niveau	Niveau
pump	pomp	Pumpe

Figure 4.8: Language dialog box.

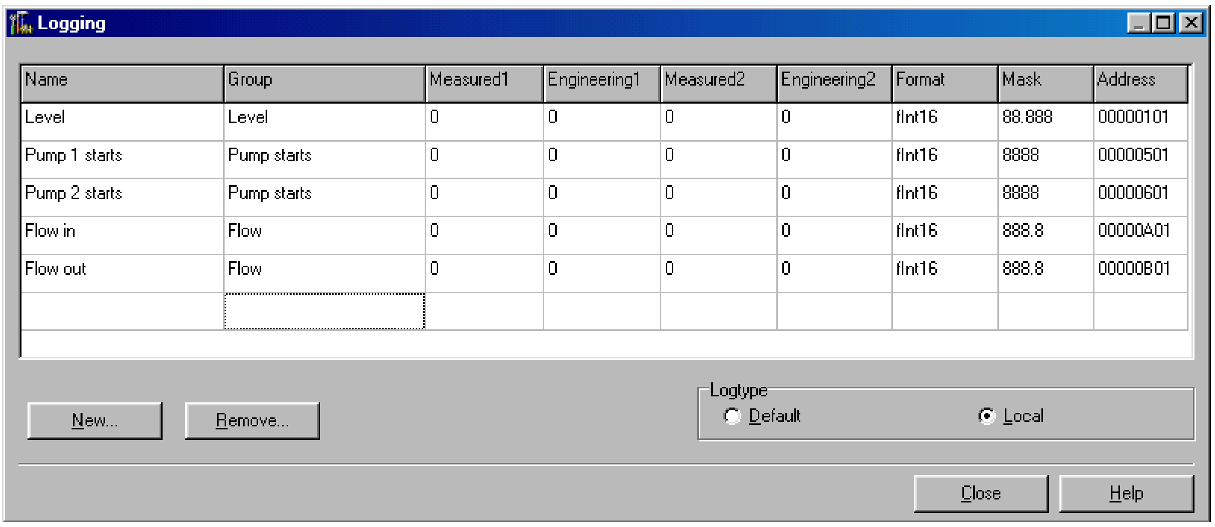
New
Click New to display the Add New Language dialog box, where you can enter a name for a new language. A new column will be added to the language grid. Now, you can start translating the native language texts (first left column) to the added language.

Remove
Click Remove to display the Remove Language dialog box, where you can select the language to remove from the language grid.

Rename
Click Rename to display the Rename Language dialog box, where you can rename the selected language.

4.11Logging

Use the logging dialog box to configure log messages. These messages will be saved when the project is saved.



Name	Group	Measured1	Engineering1	Measured2	Engineering2	Format	Mask	Address
Level	Level	0	0	0	0	fInt16	88.888	00000101
Pump 1 starts	Pump starts	0	0	0	0	fInt16	8888	00000501
Pump 2 starts	Pump starts	0	0	0	0	fInt16	8888	00000601
Flow in	Flow	0	0	0	0	fInt16	888.8	00000A01
Flow out	Flow	0	0	0	0	fInt16	888.8	00000B01

Logtype: Default [✓ Local]

Figure 4.9: Logging dialog box.

New
Click new to add a new line.

Remove
Removes the selected line.

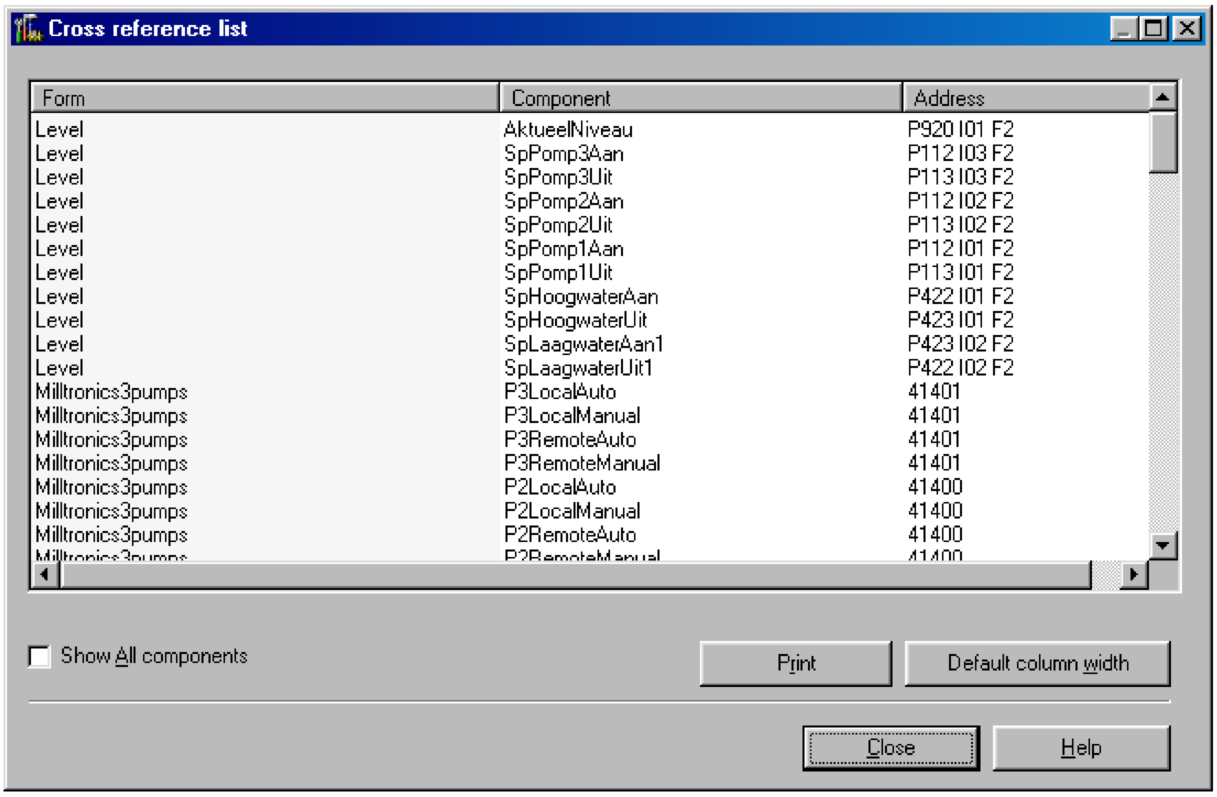
Logtype
This option is only available if the device, in project options, is CARS-unit.
Specifies the kind of logging. If the Logtype is Local then the CARS-unit will store the data. CARS will read the data. If the Logtype is default the unit will not store the data, CARS gets his data direct from the device address.

Property	Description
Name	Specifies the name of the log item.
Group	Specifies the name of the group this item belongs to.
Measured1	Specifies the first measured raw value sample.
Engineering1	Specifies the corresponding engineering value of Measured1.
Measured2	Specifies the second measured raw value sample.
Engineering2	Specifies the corresponding engineering value of Measured2.
Format	Specifies the format of the data.
Mask	Specifies the mask that represents what text is valid for this item.
Address	Specifies the device address.

Tabel 4.1 : Log properties

4.12Cross reference

The cross reference list can be used to get an overview of all the components in the selected project. You can also have an overview of the components that have an address, to check if they are right addressed.



Form	Component	Address
Level	AktueelNiveau	P920 I01 F2
Level	SpPomp3Aan	P112 I03 F2
Level	SpPomp3Uit	P113 I03 F2
Level	SpPomp2Aan	P112 I02 F2
Level	SpPomp2Uit	P113 I02 F2
Level	SpPomp1Aan	P112 I01 F2
Level	SpPomp1Uit	P113 I01 F2
Level	SpHoogwaterAan	P422 I01 F2
Level	SpHoogwaterUit	P423 I01 F2
Level	SpLaagwaterAan1	P423 I02 F2
Level	SpLaagwaterUit1	P422 I02 F2
Milltronics3pumps	P3LocalAuto	41401
Milltronics3pumps	P3LocalManual	41401
Milltronics3pumps	P3RemoteAuto	41401
Milltronics3pumps	P3RemoteManual	41401
Milltronics3pumps	P2LocalAuto	41400
Milltronics3pumps	P2LocalManual	41400
Milltronics3pumps	P2RemoteAuto	41400
Milltronics3pumps	P2RemoteManual	41400

Figure 4.10: Cross refrence dialog box.

Show All components
Here you can select all the componts or only the components that have an address.

Print
Click print to print the list.

Default column width
Will resize the column’s to the default size.

5 Working with the Form designer

Forms and components are the foundation of a CARS Builder application. You design the user interface for your application using forms. Forms can contain buttons, edit boxes, pumps, logging components or any other components available from the Component library. When you finish designing a form, you can add it to the Form library so you can reuse it in other projects.

5.1Creating an application

In brief, to create a CARS Builder application, you do the following:

1. Start with a form.
2. Put components on the form
3. Set the properties of the components
4. Save the project

A CARS builder application usually contains multiple forms. A main form, which is the primary user interface, and other forms such as setpoint dialog boxes.

You develop your application by customizing the main form and adding and customizing other forms. You customize forms by adding components and setting their properties.

5.2Creating a form

CARS Builder provides several ways to create a new form, reuse existing forms, and customize existing forms. The user can create a new form by:

- Starting CARS Builder, which generates a blank form.
- Creating a new application by choosing New Project from the File menu. CARS Builder generates an new form whenever it creates a new project.
- Adding a new form to an existing project by choosing Add Form from the Project menu.

After the form is created, the user can add components to the form.

5.3Manipulating components in forms

You can select, cut, copy, paste, move and delete components the same way you do in other Windows applications. Some skills may be specific to CARS Builder including:

- Adding components to a form
- Aligning components

Placing components on a form

To add a component to the center of a form, you have to double-click the component in the Component palette. If components already reside in the center of the form, new ones are placed on top. You can move them to the desired position.

You can add components to a specific loaction in the form. You have to click the component on the Component palette and move the cursor to where you want the upper left corner of the component to appear in the form and the click the form. The component appears in its default size at the position you clicked on the form.

You can also place multiple copies of the same component on the form by holding down the Shift key. You have to click the form once for each copy you want of the component. Clicking on the form continues to add the component to the form, as long as the component remains selected in the palette. You have to click the pointer icon to clear the selected component.

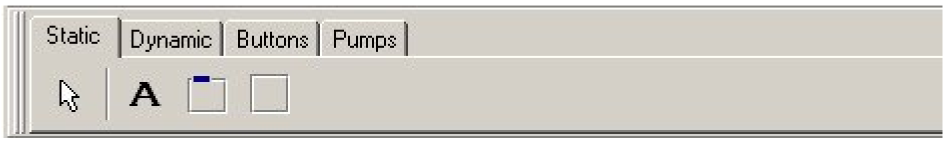
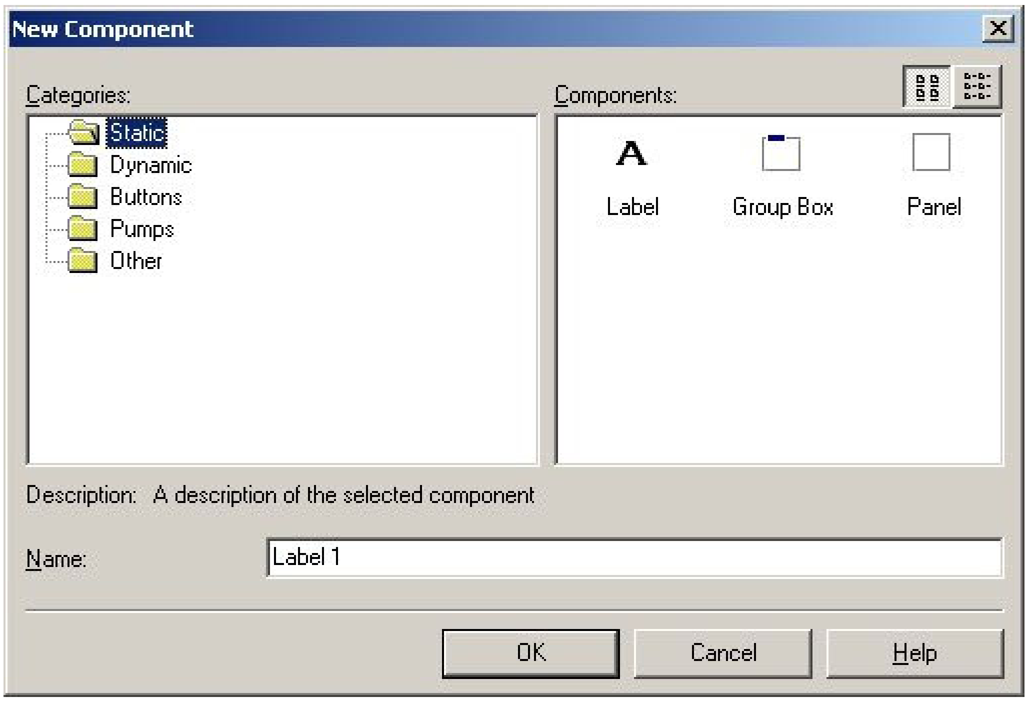


Figure 5.1: Component palette.

Choose Add Component form the Project menu to add a component that is not available on the Component palette. The New Component dialog box appears. Select a component from the Components list and click OK. Move the cursor to where you want the upper left corner of the component to appear in the form and the click the form. The component appears in its default size at the position you clicked on the form.



Description: A description of the selected component

Name: Label 1

Figure 5.2: New Component dialog box.

Aligning components

You can use the Layout toolbar or the Format menu to set the alignment of selected components. When aligning a group of components, the first component you select is used to which the other components are aligned. You can continue to choose or modify alignment options as long as the components remain selected.

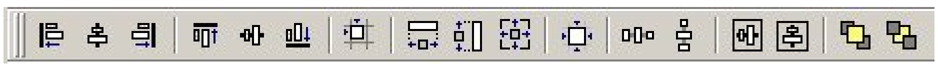


Figure 5.3: Layout toolbar.
You can also align components using the grid. The evenly spaced dots that appear in the form are the form grid. The grid makes it easier to align components visually. By default, both the grid and its Snap to grid option, which causes the left and top sides of each component to always align with the nearest grid markings, are enabled. You can change the grid size. Select Options from the Tools menu to set grid options.

Once you have aligned the components on a form, you can prevent components form being moved accidentally by locking the position of the components. Select Lock Controls from the Format menu to lock the position of the components on a form.

5.4Setting component properties and events

You can set component properties and events using the Component inspector. The Component selector at the top of the Component inspector is a drop-down list containing all the components on the active form. This lets you quickly select different components on the current form.

Properties page

The Properties page of the Component Inspector enables you to set design-time properties for components on youre form, and for the form itself.

Events page

The Events page of the Component Inspector enables you to connect forms and components to program events. When you click an event from the Events page, CARS Builder creates an event handler and switches focus to the script editor. In the script editor, you write the script inside event handlers that specifies how a component or form responds to a particular event.

The Events page displays only the events of the component that is slected in the form.
For more about script, see the chapter about script.

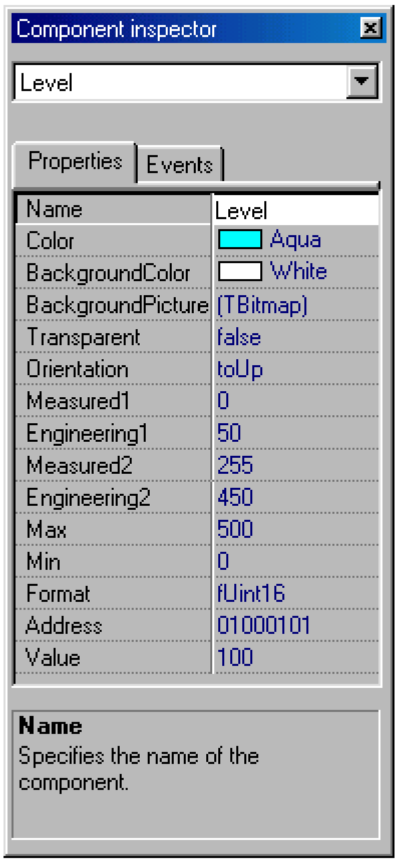
How the Component inspector displays properties

The Properties page below the Component selector enables you to set properties for components on your form, and for the form itself. The Component inspector dynamically changes the set of properties it displays, based on the component selected. The first column on the Properties page lists the names of the selected component's properties. The second column lists the property values.

To change a component property, you follow these steps:

1. Select the component in the form or with the Component selector.
2. Select the property that you want to change by selecting it from the Properties page.
3. Enter a new value for that property.

Figure 5.4: Component inspector dialog box.



Component inspector

Level

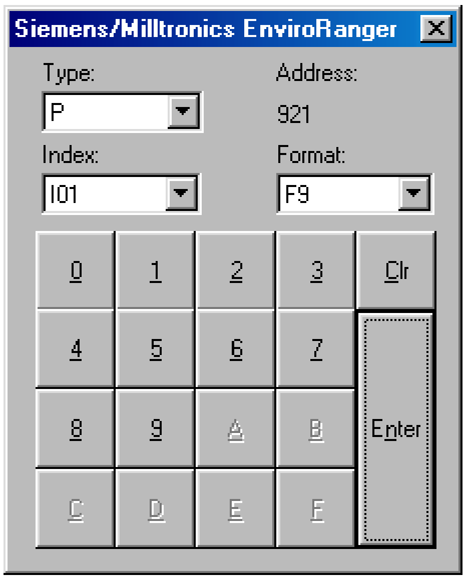
Properties | Events

Name	Level
Color	🟦 Aqua
BackgroundColor	⬜ White
BackgroundPicture	(TBitmap)
Transparent	false
Orientation	toUp
Measured1	0
Engineering1	50
Measured2	255
Engineering2	450
Max	500
Min	0
Format	fUint16
Address	01000101
Value	100

Name
Specifies the name of the component.

5.5AdressEditor

With the address editor you fill in the device address of a component. The showed and enbaled items will be depend on the type of device.



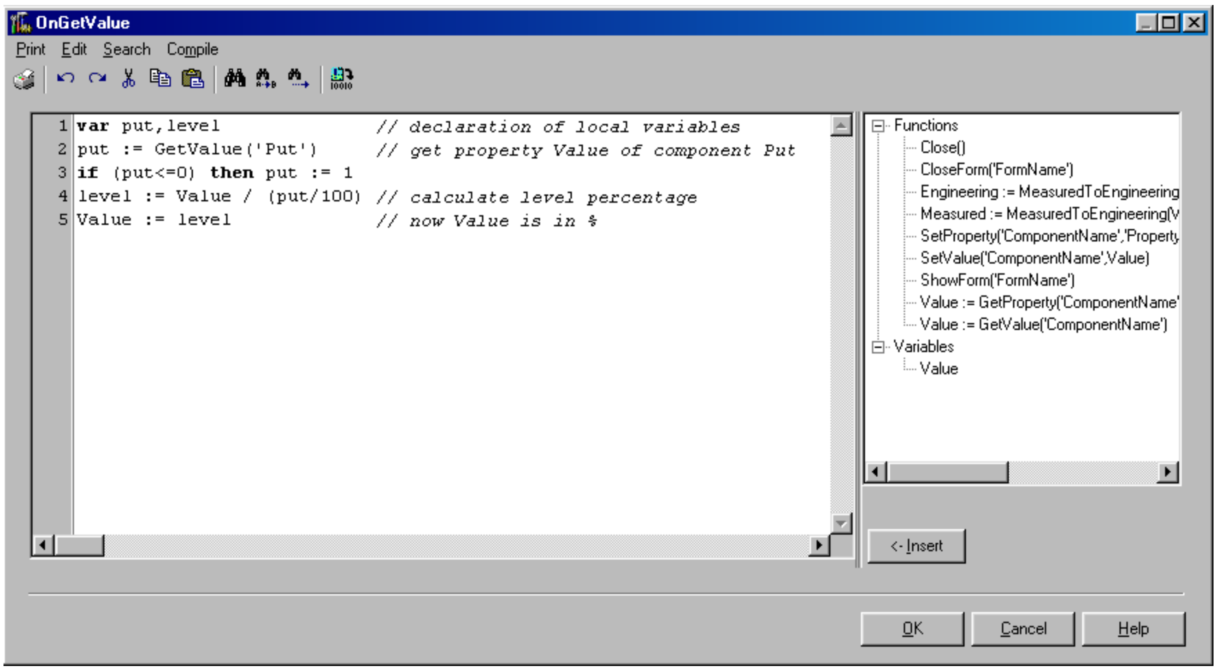
Siemens/Milltronics EnviroRanger

Type: P
Address: 921
Index: 101
Format: F9

Figure 5.5: Address editor dialog box.

5.6SriptEditor

With the script editor you can add some script code to a component.



```
var put,level              // declaration of local variables
put := GetValue('Put')     // get property Value of component Put
if (put<=0) then put := 1
level := Value / (put/100) // calculate level percentage
Value := level             // now Value is in %
```

```
Functions
  Close()
  CloseForm('FormName')
  Engineering := MeasuredToEngineering
  Measured := MeasuredToEngineering(V
  SetProperty('ComponentName','Property
  SetValue('ComponentName',Value)
  ShowForm('FormName')
  Value := GetProperty('ComponentName'
  Value := GetValue('ComponentName')
Variables
  Value
```

Figure 5.6: Script editor dialog box.

Insert
Click on this button or dubble click on the functie name or variable name, will add it to the script code.

Print
Will print the scriptcode.

Undo
Will undo the change.

Redo
Will redo the change.

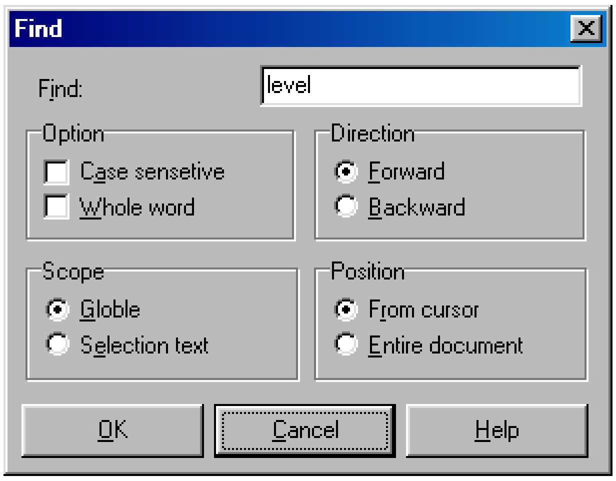
Cut
Will cut the selected text.

Copy
Will copy the selected text.

Paste
Will paste from clipboard.

Find

With this function you can find a specific text in the script code.



Find: level

Option
[ ] Case sensetive
[ ] Whole word

Direction
[✓ Forward] Backward

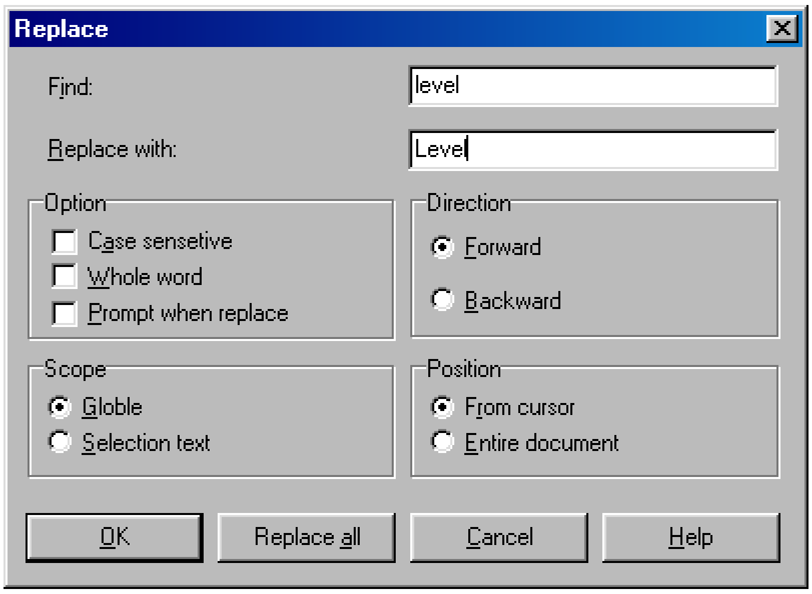
Scope
[✓ Globle] Selection text

Position
[✓ From cursor] Entire document

Figure 5.7: Find dialog.

Replace

With this function you can replace a specific text by an other text in the script code.



Find: level

Replace with: Level

Option
[ ] Case sensetive
[ ] Whole word
[ ] Prompt when replace

Direction
[✓ Forward] Backward

Scope
[✓ Globle] Selection text

Position
[✓ From cursor] Entire document

Figure 5.8: Replace dialog.

Search again

Do the search action again.

Compile

Compiles the scriptcode to check if there are errors.

5.7Creating Component templates

Component templates are components that have been configured to have a user-defined set of property values. You can add your own defined component to the Component Library by creating a component template. Select Create Component Template form the Project menu to add the current selected component to the Component Library. The Create Component Template dialog box appears.

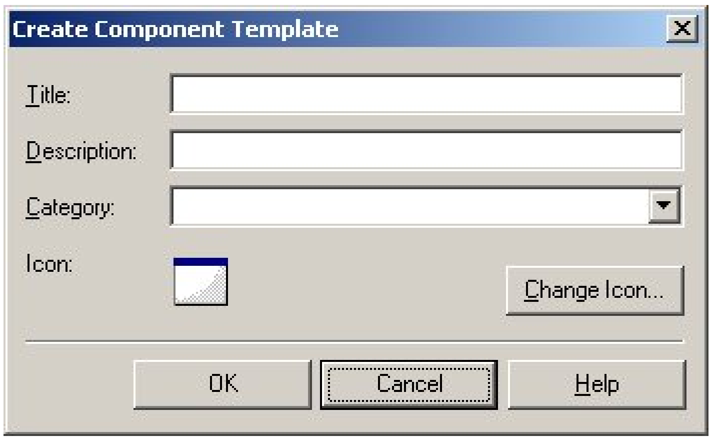


Figure 5.7: Create Component Template dialog box.

Title
Specify a template name for the currently selected component.

Description
Enter a brief description for the component template.

Category
Choose a category in which the component should appear in the New Component dialog box. If you specify a category that does not exist, a new category is created.

Icon
To specify an icon for the component template, choose the Change Icon button. Navigate the Open Icon dialog box to find a directory of images, and choose a new bitmap for the icon. The bitmap you choose must be no larger than 32 pixels by 32 pixels.

The next time you choose Add Component from the Project menu and click the Category you selected above, your component appears in the Components list, with the icon and title you chose. You can also add the component to the Component palette by selecting Customize Component Palette from the Tools menu.

5.8Customizing the Component library

Use the Customize Component Library dialog box to customize the Component library. You can add, rename, or remove categories and component templates. You can access this dialog box by choosing Customize Component Library from the Tools menu.

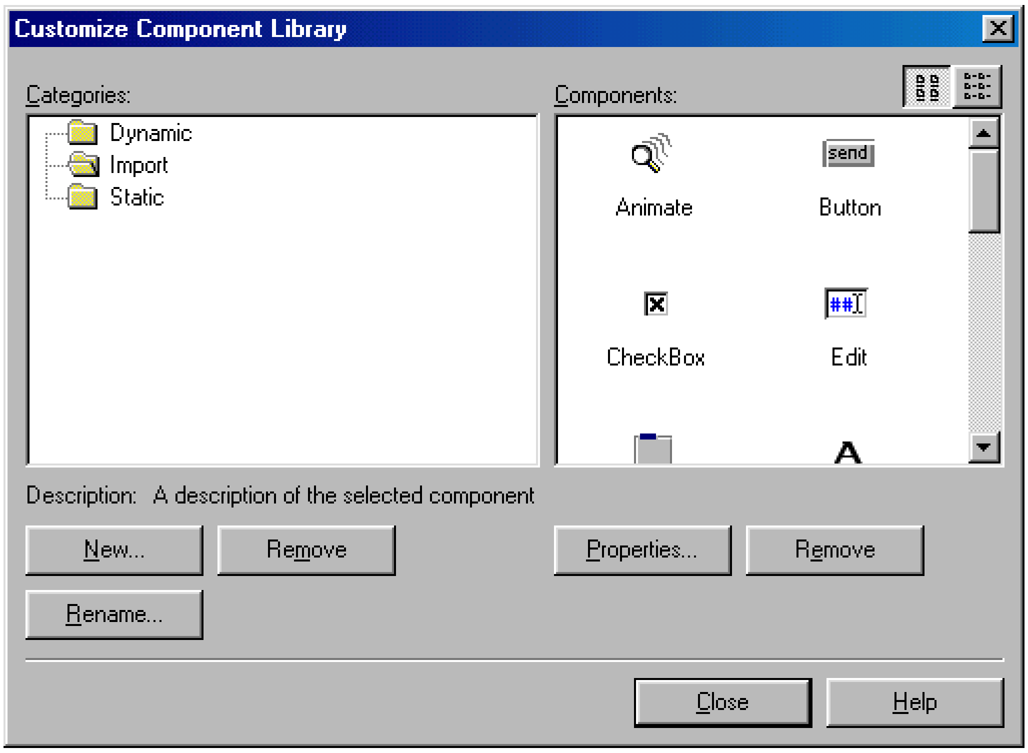


Figure 5.6: Customize Component Library dialog box.

Categories
Lists the categories in the Component library. You can rearrange these categories or view and rearrange their component templates in the Components list.

Components
Lists the component templates of the currently selected category in the Categories list.

Large Icons
Displays component templates as large icons with a title below it.

List
Displays component templates as small icons with a title to the right of it.

Use the following buttons when an item is selected in the Categories list.

New
Click New to display the Add New Category dialog box, where you can create a category. Once you have created a new category, you can move component templates from other categories into it or add new component templates to it.

Remove
Removes the selected category from the list. Before you can delete a category, it must be empty of component templates.

Rename
Click Rename to display the Rename Category dialog box, where you can rename the selected category.

Use the following buttons when an item is selected in the Components list.

Properties
Click Properties to display the Component Template Properties dialog box, where you can change the title, description, category and icon of the component template.

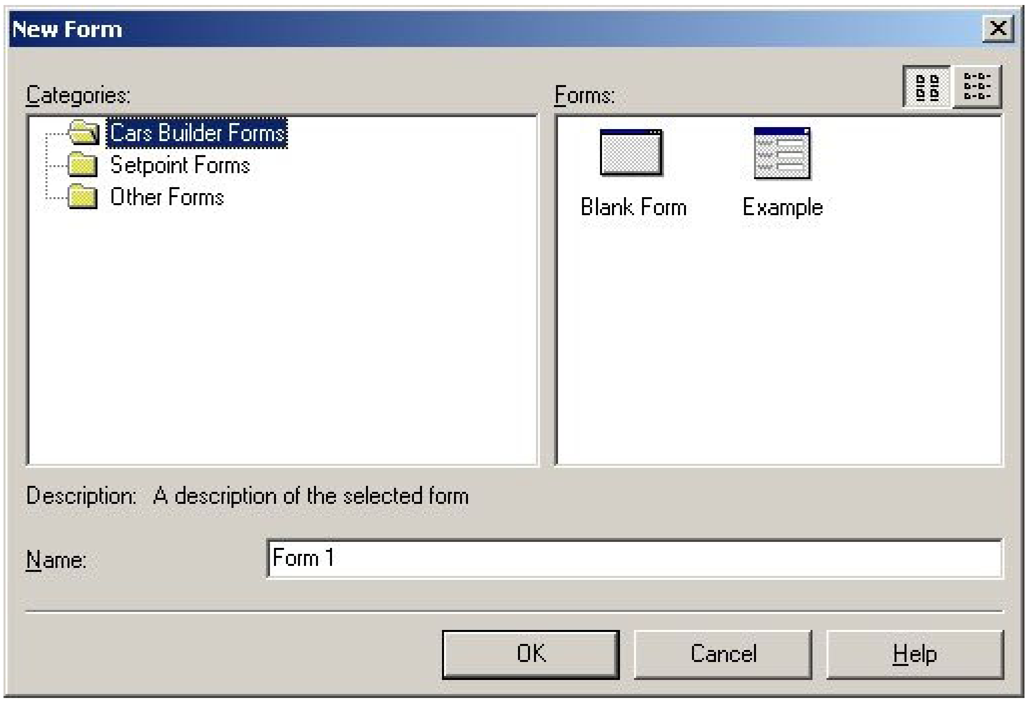
Remove
Removes the selected component template from the Component Library.

5.9Using predesigned forms

When you start CARS Builder, it opens with an empty project consisting of a single, blank form that contains no components. You can then place components on the form. You can also choose to add a predesigned form to the project and use it or modify it.

Adding an existing form

You can use any of the predesigned forms from the Form library in your application. To add a predesigned form to your project, choose Add Form from the Project menu. The New Form dialog box appears. When you select a form from the Categories list, CARS Builder adds the form to the project you have open. You can now use this form the way you would any form in a project.



Description: A description of the selected form

Name: Form 1

Figure 5.7: New Form dialog box.

Creating form templates

You can add your own form to the Form Library by creating a form template. Select Create Form Template form the Project menu to add the current form to the Form Library. The Create Form Template dialog box appears.

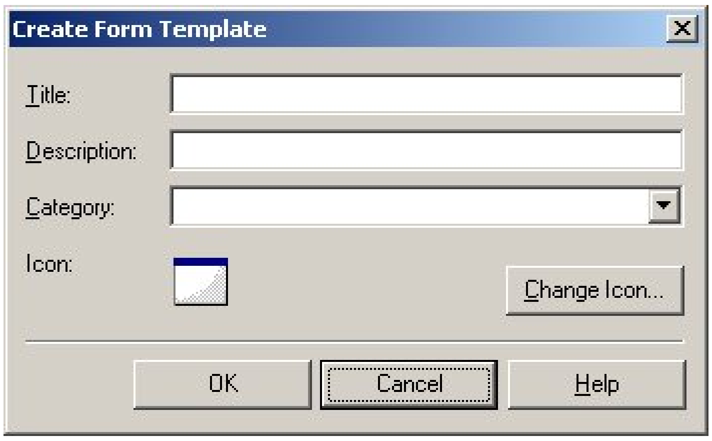


Figure 5.8: Create Form Template dialog box.

Title
Specify a template name for the currently selected form.

Description
Enter a brief description for the form template.

Category

Choose a category in which the form should appear in the New Form dialog box. If you specify a category that does not exist, a new category is created.

Icon

To specify an icon for the form template, choose the Change Icon button. Navigate the Open Icon dialog box to find a directory of images, and choose a new bitmap for the icon. The bitmap you choose must be no larger than 32 pixels by 32 pixels.

The next time you choose Add Form from the Project menu and click the Category you selected above, your form appears in the Forms list, with the icon and title you chose.

5.10Customizing the Form library

Use the Customize Form Library dialog box to customize the Form library. You can add, rename, or remove categories and form templates. You can access this dialog box by choosing Customize Form Library from the Tools menu.

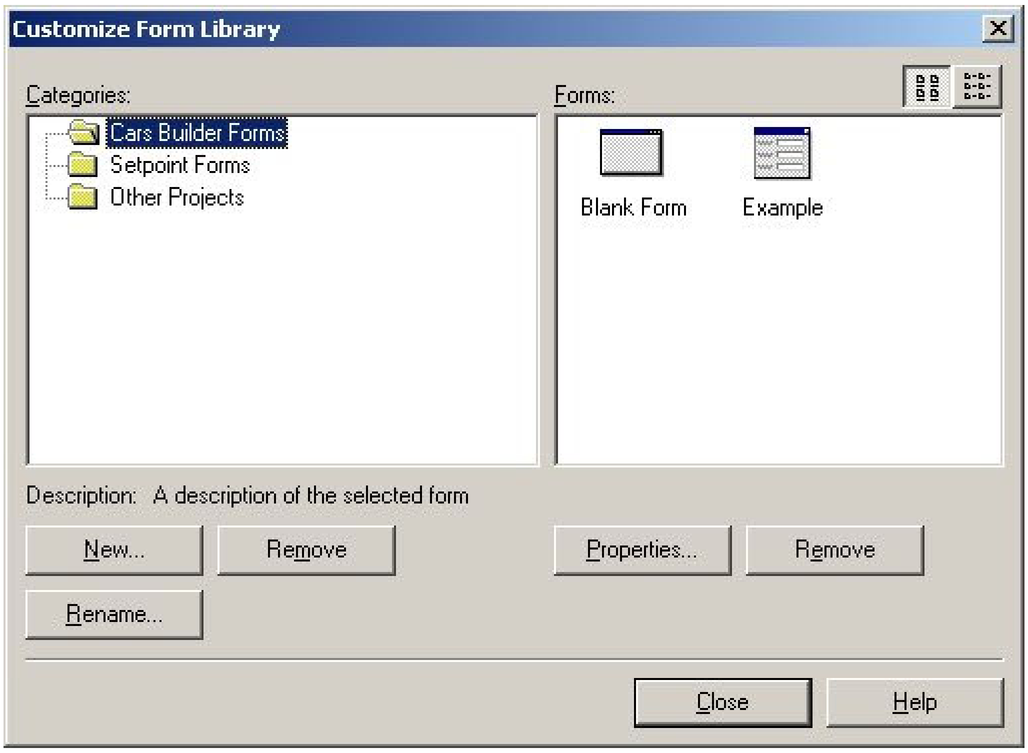


Figure 5.9: Customize Form Library dialog box.

Categories

Lists the categories in the Form library. You can rearrange these categories or view and rearrange their form templates in the Forms list.

Forms

Lists the form templates of the currently selected category in the Categories list.

Large Icons

Displays form templates as large icons with a title below it.

List

Displays form templates as small icons with a title to the right of it.

Use the following buttons when an item is selected in the Categories list.

New
Click New to display the Add New Category dialog box, where you can create a category. Once you have created a new category, you can move form templates from other categories into it or add new form templates to it.

Remove
Removes the selected category from the list. Before you can delete a category, it must be empty of form templates.

Rename
Click Rename to display the Rename Category dialog box, where you can rename the selected category.

Use the following buttons when an item is selected in the Forms list.

Properties
Click Properties to display the Form Template Properties dialog box, where you can change the title, description, category and icon of the form template.

Remove
Removes the selected form template from the Form Library.

6 Customizing the environment

6.1External Tools

You can use the External Tools dialog box to add external tools, such as CARS 6 or Paint to the Tools menu. Adding external tools allows you to easily launch other Windows applications while working in the development environment. You can specify arguments and a working directory when launching the tool. The External Tools dialog box is available on the Tools menu.

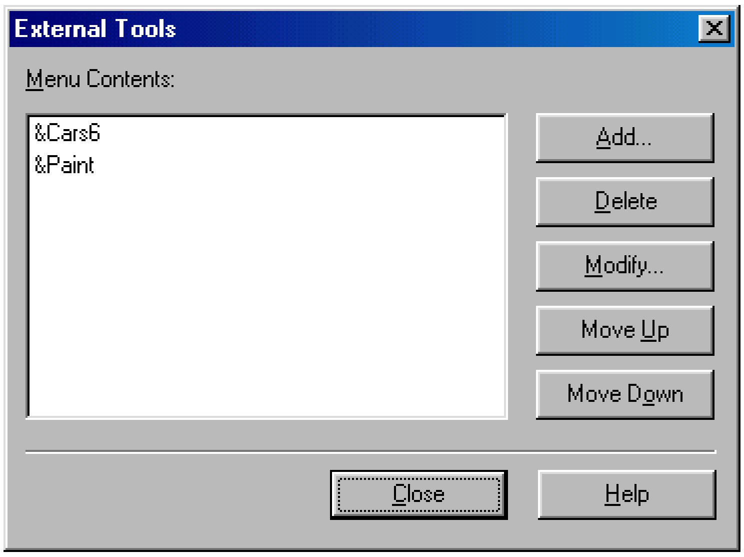


Figure 6.1: External Tools dialog box.

Menu contents
Lists the titles of the items currently added to the Tools menu. Use the Move up and Move down buttons to change the order the items that appear on the menu. Use the Delete button to remove an item from the menu.

Add
Clears the text boxes so you can specify a new tool.

Delete
Removes the tool from the Menu contents list as well as from the Tools menu.

Modify
Shows figure 6.2 to modify the tool properties.

Move up
Moves the selected tool higher in the list of tools that appear on the Tools menu.

Move down
Moves the selected tool lower in the list of tools that appear on the Tools menu.

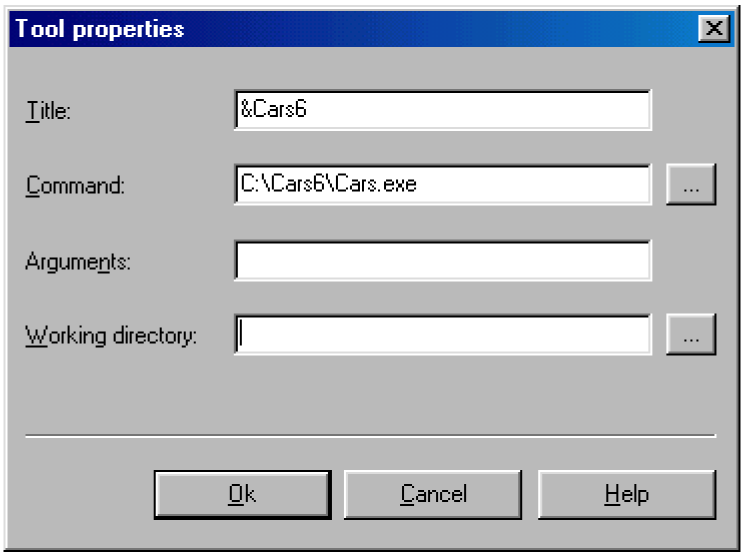


Figure 6.2: Tool properties dialog box.

Title
The name of the tool that will appear on the External Tools sub menu of the Tools menu. Place an ampersand before a letter in the name of the tool to use that letter as an accelerator key for the tool. For example, &CARS 6 would display CARS 6 on the Tools menu.

Command
Specifies the path to the .exe, .com or other file that you intend to launch.

Arguments
Specifies the command line switches that are passed to the tool when the tool is selected on the menu.

Working directory
Specifies the working directory of the tool.

6.2Customizing the Component palette

Use the Customize Component Palette dialog box to customize the way the Component palette appears. You can add, rename, or remove pages and components. You can access this dialog box by choosing Customize Component Palette from the Tools menu.

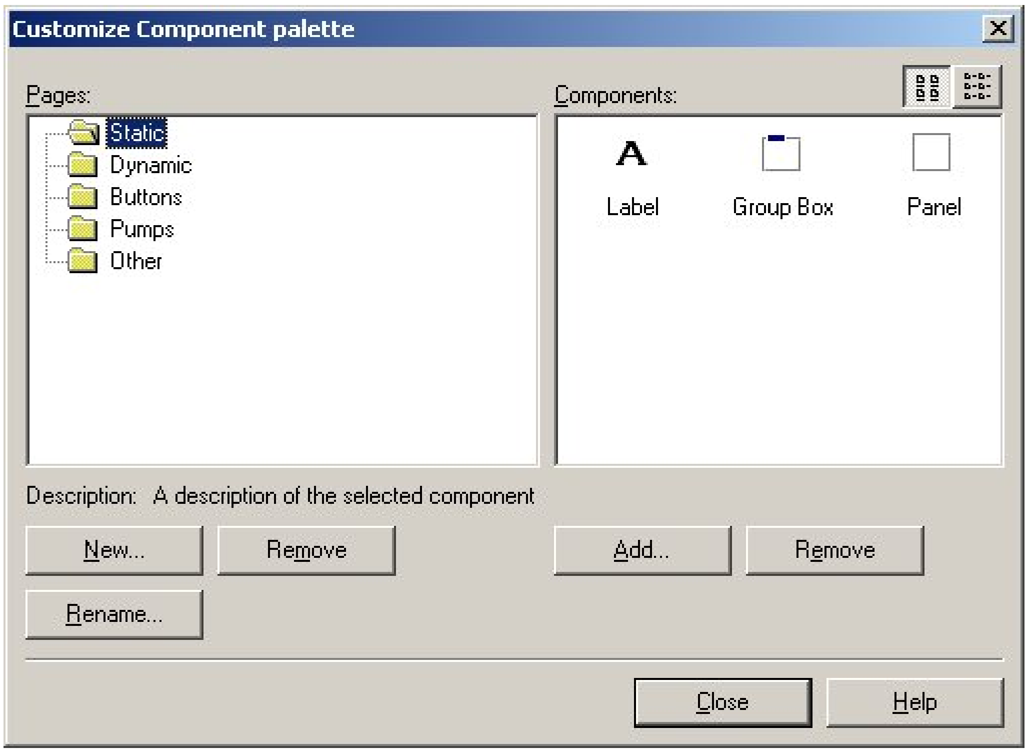


Figure 6.3: Customize Component palette dialog box.

Pages
Lists the pages in the Component palette. You can rearrange these pages or view and rearrange their components in the Components list.

Components
Lists the components on the currently selected page in the Pages list.

Large Icons
Displays components as large icons with a label below it.

List
Displays components as small icons with a label to the right of it.

Use the following buttons when an item is selected in the Pages list.

New
Click New to display the Add New Page dialog box, where you can create new pages on the Component palette. Once you have created a new Component palette page, you can move components from other pages onto it or add new components to it.

Remove
Removes the selected page from the palette. Before you can delete a page, it must be empty of components.

Rename
Click Rename to display the Rename Page dialog box, where you can rename the selected page.

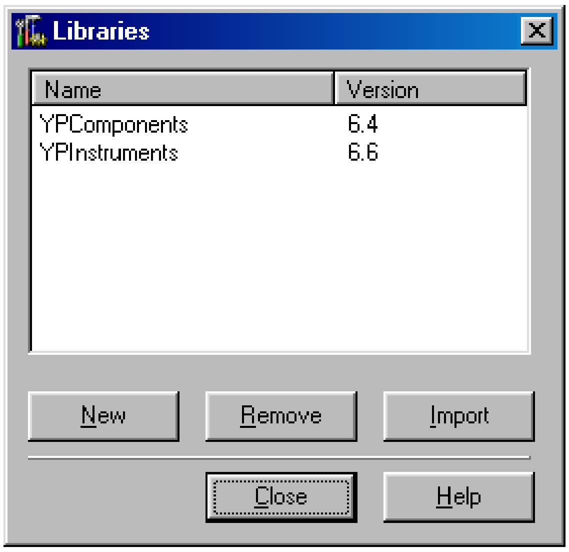
Use the following buttons when an item is selected in the Components list.

Add
Click Add to display the Add Component dialog box, where you can select a component to add to the selected Page.

Remove
Removes the selected component from the Component palette.

6.3Import library

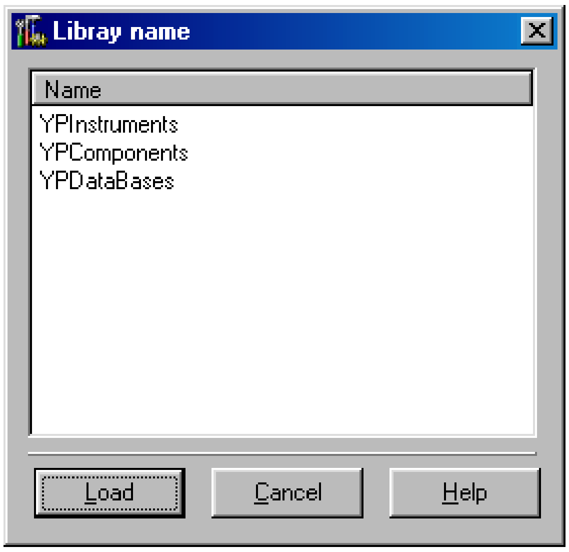
Here you can add/remove libraries and import components from the libraries. Some of this functions are only possible if the is no project loaded. If a function is not possible the button is disabled.
It gives you als on overview of the libraries loaded.


Libraries

Name	Version
YPComponents	6.4
YPInstruments	6.6

Figure 6.4: Libraries dialog box.

New
Add a new library to CARS Builder



YPInstruments
YPComponents
YPDataBases

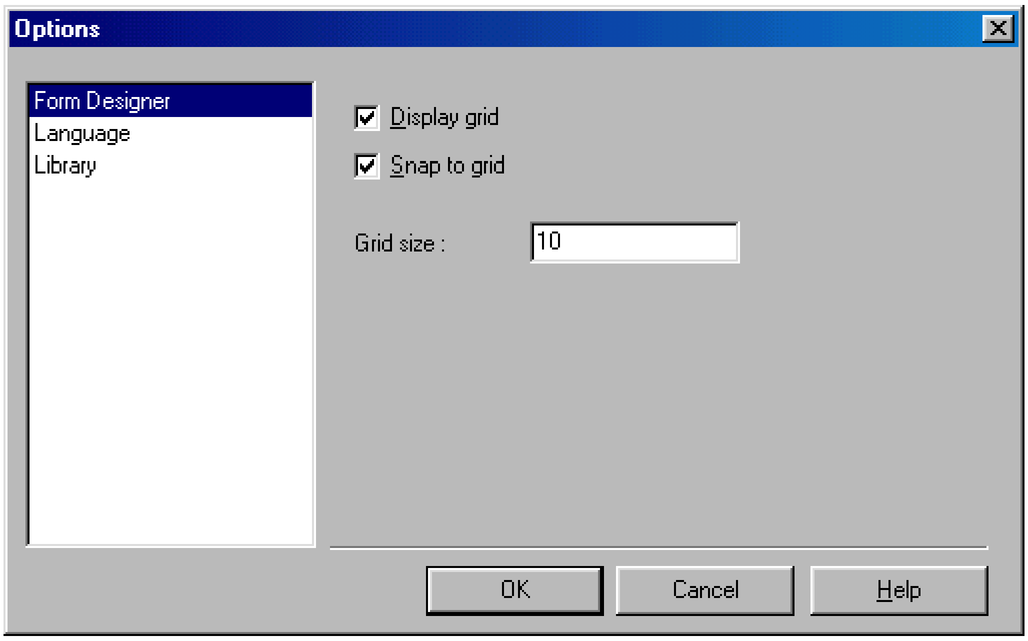
Figure 6.5: Library name dialog box.

Remove
Removes the seceted library from CARS Builder.

Import
Import the components in CARS Builder from the selected library. The components will be stored in the template directory with subdirectory the name of the library.

6.4Setting environment options

You can change many of the settings that affect the development environment in the Options dialog box. The Options dialog box is divided into two parts: a navigation pane on the left and a page display area on the right. The navigation pane contains a variety of pages, such as Form Designer, Language and Library. When you select a page, its contents appear in the page display area. To access this dialog box select Options from the Tools menu.



☑ Display grid
☑ Snap to grid
Grid size : 10

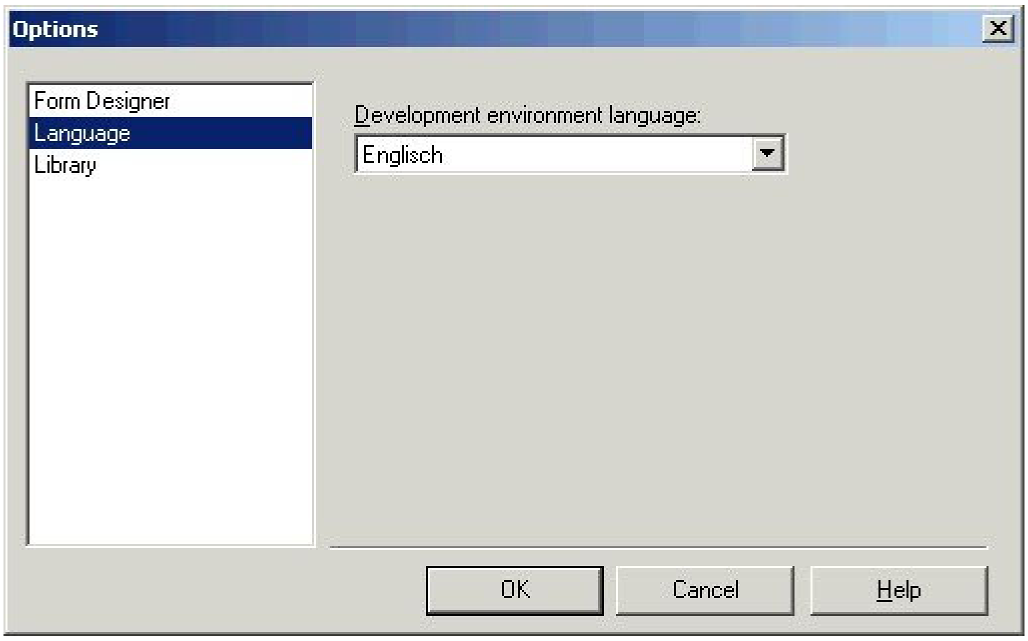
Figure 6.4: Options dialog box for Form Designer settings.

Form Designer
Use this page to set grid preferences that make it easier to design forms.

Display grid
Displays dots on the form to make the grid visible.

Snap to grid
When components are added or moved on a form, they are automatically aligned with the nearest guidelines in the positioning grid when you release the mouse.

Grid size
Sets grid spacing in pixels along the axis. Specify a higher number (between 2 and 128) to increase grid spacing. The default value is 8 pixels.

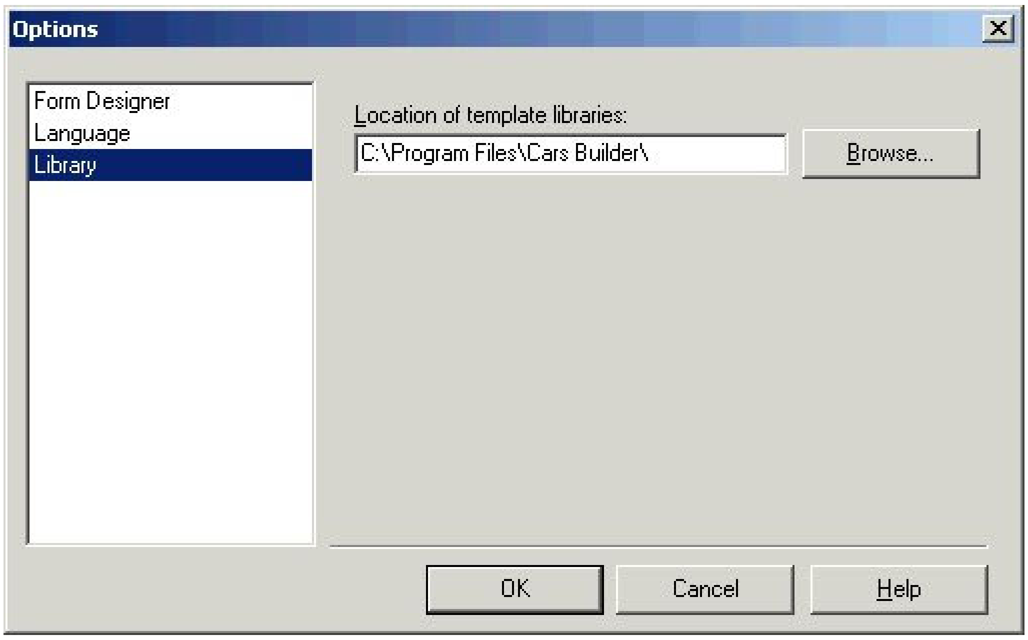


Development environment language: Englisch

Figure 6.5: Options dialog box for Language settings.

Language
Use this page to set the development environment language.

Development environment language
Lists the languages available for the development environment. Select a language to be used for the menu's and dialog boxes in the environment.



Location of template libraries: C:\Program Files\Cars Builder\

Figure 6.6: Options dialog box for Library settings.

Library
Use this page to set the directory for the template libraries.

Location of template libraries
Specifies the location where CARS Builder looks for the Project, Form and Component template files. If this field is empty, CARS Builder looks for the templates in the home directory of CARS Builder.

7 Using the help system

The Help system provides online access to detailed information about CARS Builder.

7.1Getting help

You can display online Help while using CARS Builder in any of the following ways:

- Choose Contents from the Help menu.
- Press the key F1.
- In a dialog box, click the Help button.

7.2Displaying online information

The Help Contents displays a Help Topics dialog box showing the contents of the User's Guide. From the contents, you can view a topic by clicking on it. You can also click the Index or Find tabs on the Help Topics dialog box to access information in a different way. The index provides entries that either display an associated topic or, if there are several topics, displays a list of relevant topics to choose from. Find provides a full text search so you can search for specific terms throughout the Help system.

7.3Displaying context sensitive information

Pressing F1 within the CARS Builder environment displays context sensitive help. If a dialog box is displayed, F1 provides information on using the dialog box and its options. Also you can select any property in the Component inspector and press F1 to see a description of it.

8 Components

This chapter describes the standard CARS Builder components. These components are available in the YPComponents.dll library file.

8.1Label

<u>**Property name**</u>	<u>**Description**</u>
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Alignment	Controls the horizontal placement of the text within the component.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
Measured1	Specifies the first measured raw value sample.
Engineering1	Specifies the corresponding engineering value of Measured1.
Measured2	Specifies the second measured raw value sample.
Engineering2	Specifies the corresponding engineering value of Measured2.
Mask	Specifies the mask that represents what text is valid for this component.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

With the property Mask you can specify the text that is valid for this component. For example, if you specify Mask as 88:88 and Value is 100 the Text property will be displayed as 01:00. If you enter a Mask that is not valid, CARS Builder will replace the none valid characters by valid characters.

<u>**Event name**</u>	<u>**Description**</u>
OnGetValue	Specifies the script that will be executed when the component receives data.

8.2TextLabel

Property name	**Description**
Name	Specifies the name of the component.
TextItems	Specifies a stringlist that identifies the component to the user, depending on the contents of value.
Alignment	Controls the horizontal placement of the text within the component.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
Format	Specifies the format of the contens of value.
Address	Specifies the device address.
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.3GroupBox

Property name	**Description**
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent's tab order.

Event name	**Description**

8.4Panel

Property name	**Description**
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Width	Specifies the horizontal size in pixels.
Height	Specifies the vertical size in pixels.
BevelInner	Determines the style of the inner bevel.
BevelOuter	Determines the style of the outer bevel.
BevelWidth	Determines the width, in pixels, of both the inner and outer bevels.
BorderStyle	Determines the style of the line drawn around the perimeter of the component.
BorderWidth	Specifies the distance, in pixels, between the outer and inner bevels.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent’s tab order.

Event name	**Description**

8.5Editbox

Property name	**Description**
Name	Specifies the name of the component.
Text	Contains a string associated with the component.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent’s tab order.
Measured1	Specifies the first measured raw value sample.
Engineering1	Specifies the corresponding engineering value of Measured1.
Measured2	Specifies the second measured raw value sample.
Engineering2	Specifies the corresponding engineering value of Measured2.
Max	Specifies the upper limit in engineering units.
Min	Specifies the lower limit in engineering units.
Mask	Specifies the mask that represents what text is valid for this component.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

With the property Mask you can specify the text that is valid for this component. For example, if you specify Mask as 88:88 and Value is 100 the Text property will be displayed as 01:00. If you enter a Mask that is not valid, CARS Builder will replace the none valid characters by valid characters.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.
OnSetValue	Specifies the script that will be executed when the component will send the data.

8.6Checkbox ☒

Property name	**Description**
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent’s tab order.
Inverse	Displays checked inverse.
Format	Specifies the format of the contens of value.
Bit	Specifies the bit number to check.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.7Radiobutton

Property name	**Description**
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent’s tab order.
Inverse	Displays checked inverse.
Format	Specifies the format of the contens of value.
Bit	Specifies the bit number to check.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**

8.8Button

Property name	**Description**
Name	Specifies the name of the component.
Text	Specifies a text string that identifies the component to the user.
Color	Specifies the background color of the component.
Font	Controls the attributes of text written on the component.
Font.Color	Specifies the color of the text.
Font.Name	Identifies the typeface of the font.
Tabstop	Determines if the user can tab to the component.
Taborder	Indicates the position of the component in its parent’s tab order.
Action	Specifies which action to do on click.
Form	Specifies which form will be showed.

Event name	**Description**

8.9Pump

Property name	**Description**
Name	Specifies the name of the component.
States	Specifies all the states of the component.
State.DefaultColor	Specifies the default color if no state is active.
State.State01	Specifies this state.
State.State01.Color	Specifies the color for this state.
State.State01.Bit	Specifies the bit for this state.
..	
..	
State.State16	Specifies this state.
State.State16.Color	Specifies the color for this state.
State.State16.Bit	Specifies the bit for this state.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
TransparentColor	Determines which color of the component is to be transparent when the component is drawn.
Orientation	Specifies the orientation of the component.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.10MilltronicsPump

Property name	**Description**
Name	Specifies the name of the component.
States	Specifies all the states of the component.
State.DefaultColor	Specifies the default color if no state is active.
State.State01	Specifies this state.
State.State01.Color	Specifies the color for this state.
State.State01.Bit	Specifies the bit for this state.
..	
..	
State.State16	Specifies this state.
State.State16.Color	Specifies the color for this state.
State.State16.Bit	Specifies the bit for this state.
Transparent	Specifies whether controls that sit below the component on a form can
be	
	seen through the component.
TransparentColor	Determines which color of the component is to be transparent when
the	
	component is drawn.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives
data.	

8.11Led

Property name	**Description**
Name	Specifies the name of the component.
States	Specifies all the states of the component.
State.DefaultColor	Specifies the default color if no state is active.
State.State01	Specifies this state.
State.State01.Color	Specifies the color for this state.
State.State01.Bit	Specifies the bit for this state.
..	
..	
State.State16	Specifies this state.
State.State16.Color	Specifies the color for this state.
State.State16.Bit	Specifies the bit for this state.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
TransparentColor	Determines which color of the component is to be transparent when the component is drawn.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.12Picture

Property name	**Description**
Name	Specifies the name of the component.
States	Specifies all the states of the component.
Sate.DefaultPicture	Specifies the default picture if no state is active.
State.State01	Specifies this state.
State.State01.Picture	Specifies the picture for this state.
State.State01.Bit	Specifies the bit for this state.
..	
..	
State.State16	Specifies this state.
State.State16.Picture	Specifies the picture for this state.
State.State16.Bit	Specifies the bit for this state.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
TransparentColor	Determines which color of the component is to be transparent when the component is drawn.
Format	Specifies the format of the contens of value.
Address	Specifies the device address.
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.13Level

Property name	Description
Name	Specifies the name of the component.
Color	Specifies the color of the level.
BackGroundColor	Specifies the background color.
BackGroundPicture	Specifies the background picture.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
Orientation	Specifies the orientation of the component.
Measured1	Specifies the first measured raw value sample.
Engineering1	Specifies the corresponding engineering value of Measured1.
Measured2	Specifies the second measured raw value sample.
Engineering2	Specifies the corresponding engineering value of Measured2.
Max	Specify the upper limit in engineering units.
Min	Specify the lower limit in engineering units.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	Description
OnGetValue	Specifies the script that will be executed when the component receives data.

8.14LevelSetpointIndicator

Property name	**Description**
Name	Specifies the name of the component.
Color	Specifies the color of the component.
Width	Specifies the horizontal size in pixels.
Height	Specifies the vertical size in pixels.
LinkTo	Specifies to which level the indicator is linked to.
Side	Specifies to which side of the level the component is docked.
Measured1	Specifies the first measured raw value sample.
Engineering1	Specifies the corresponding engineering value of Measured1.
Measured2	Specifies the second measured raw value sample.
Engineering2	Specifies the corresponding engineering value of Measured2.
Format	Specifies the format of the contens of value.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.15Animation

Property name	**Description**
Name	Specifies the name of the component.
Color	Specifies the background color of the component.
Pictures	Specifies the pictures for this component.
Pictures.DefaultPicture	Specifies the picture when the animation is not active.
Pictures.Picture01	Specifies a picture.
..	..
Pictures.Picture16	Specifies a picture.
Interval	Specifies the interval time in msec.
Transparent	Specifies whether controls that sit below the component on a form can be seen through the component.
Format	Specifies the format of the contens of value.
Bit	Specifies the bit number to check.
Address	Specifies the device address
Value	Specifies the raw value of the device address.

Event name	**Description**
OnGetValue	Specifies the script that will be executed when the component receives data.

8.16TimeSetpoint

This component will be used to write the time of the main station in the local device of the station. The time will be written once if the MainForm of the station appears.

Property name	**Description**
Name	Specifies the name of the component.
Mask	Specifies the mask that represents the data that is send to the device.
DataFormat	Specifies the dataformat of the component.
Address	Specifies the device address.

Event name **Description**

Mask characters:

X	= Don’t care.
K	= Wil be used to set the clock. The value 1 will be written on the address.
W	= Day of the week.
D	= Day.
M	= Month.
C	= Century.
Y	= Year.
H	= Hour.
m	= Minute.
s	= Second.

Each mask character will define a byte in the address.

Example:
```
Addresss       = 0100C006
Mask           = XKCYMDXWHmXs
             X   K  C Y   M D  X W  H m  X s
0100C006 0001 2002 1128 006 1450 0042
```

8.17ComponentList

This component can be used to make a list of components wich have some special functions.

Property name	Description
Name	Specifies the name of the component.
ComponentList	Specifies the components that can be used as components with a special function.

Event name	Description

9 Forms

This chapter describes the standard CARS Builder forms. These forms are available in the YPComponents.dll library file.

9.1Form

Property name	Description
Name	Specifies the name of the component.
Caption	Specifies a text string that identifies the form to the user (Titlebar).
Color	Specifies the color of the form.

Event name	Description

10 Scripter

With the scripter you can more customize the component. You can write code in a pascal-like langauge. This language is a subset of standard pascal, but it also extends some features to simplify script-building process.

The scripter has the following features:

- Run-time Pascal language interpreter
- Supports try..except and try..finally blocks in script
- Allows reading/writing of variables and reading constants in script
- Most of Delphi system procedures (conversion, date, formatting, string-manipulation) are already included (IntToStr, FormatDateTime, Copy, Delete, etc.)

Scripter executes scripts written in Pascal syntax. Current Pascal syntax supports:

- begin .. end constructor
- procedure and function declarations
- if .. then .. else constructor
- for .. to .. do .. step constructor
- while .. do constructor
- repeat .. until constructor
- try .. except and try .. finally blocks
- case statements
- array constructors (x:=[1, 2, 3];)
- ^ , * , / , and , + , - , or , <> , >=, <= , = , > , < , div , mod , xor , shl , shr operators
- access to object properties and methods (ObjectName.SubObject.Property)

Like in pascal, statements should be terminated by ";" character. Begin..end blocks are allowed to group statements.

Identifiers:
Identifier names in script (variable names, function and procedure names, etc.) follow the most common rules in pascal : should begin with a character (a..z or A..Z), or '_', and can be followed by alphanumeric chars or '_' char. Cannot contain any other character os spaces.

Valid identifiers:
VarName
_Some
V1A2
_____Some____

Invalid identifiers:
2Var
My Name
Some-more
This,is,not,valid

Assign statements:
Just like in Pascal, assign statements (assign a value or expression result to a variable or object property) are built using ":=". Examples:

```
MyVar:=2;
Button.Caption:='This ' + 'is ok.';
```

Character strings:
Strings (sequence of characters) are declared in pascal using single quote (') character. Double quotes (") are not used.
You can also use #nn to declare a character inside a string. There is no need to use '+' operator to add a character to a string. Some examples:
```
A:='This is a text';
Str:='Text '+'concat';
B:='String with CR and LF char at the end'#13#10;
C:='String with '#33#34' characters in the middle';
```

Comments:
Comments can be inserted inside script. You can use // chars or (* *) or { } blocks. Using // char the comment will finish at the end of line.
```
//This is a comment before ShowMessage
ShowMessage('Ok');
(* This is another comment *)
ShowMessage('More ok!');
{ And this is a comment
with two lines }
ShowMessage('End of okays');
```

Variables:
You have to declare variables just using var directive and its name.
There will raise a compile error if variable is used but not declared in script. Examples:

SCRIPT 1:
```
procedure Msg;
var S;
begin
  S:='Hello world!';
  ShowMessage(S);
end;
```

SCRIPT 2:
```
var A;
begin
  A:=0;
  A:=A+1;
end;
SCRIPT 3:
var S;
  S:='Hello World!';
  ShowMessage(S);
end;
```

Indexes:
Strings, arrays and array properties can be indexed using "[" and "]" chars. For example, if Str is a string variable, the expression Str[3] returns the third character in the string denoted by Str, while Str[I + 1] returns the character immediately after the one indexed by I. More examples:
```
MyChar:=MyStr[2];
MyStr[1]:='A';
MyArray[1,2]:=1530;
Lines.Strings[2]:='Some text';
```

Arrays:
Script support array constructors and support to variant arrays. To construct an array, use "[" and "]" chars. You can construct multi-index array nesting array constructors. You can then access arrays using indexes. If array is multi-index,separate indexes using ",".
If variable is a variant array, script automatically support indexing in that variable. A variable is a variant array is it was assigned using an array constructor, if it is a direct reference to a Delphi variable which is a variant array (see Delphi integration later) or if it was created using VarArrayCreate procedure.
Arrays in script are 0-based index. Some examples:

```
NewArray := [ 2,4,6,8 ];
Num:=NewArray[1]; //Num receives "4"
MultiArray := [ ['green','red','blue'] , ['apple','orange','lemon'] ];
Str:=MultiArray[0,2]; //Str receives 'blue'
MultiArray[1,1]:='new orange';
```

If statements:
There are two forms of if statement: if...then and the if...then...else. Like normal pascal, if the if expression is true, the statement (or block) is executed. If there is else part and expression is false, statement (or block) after else is execute.
Examples:

```
if J <> 0 then Result := I/J;
if J = 0 then Exit else Result := I/J;
if J <> 0 then
begin
  Result := I/J;
  Count := Count + 1;
end
else
  Done := True;
```

While statements:
A while statement is used to repeat a statement or a block, while a control condition (expression) is evaluated as true.
The control condition is evaluated before the statement. Hence, if the constrol condition is false at first iteration, the statement sequence is never executed. The while statement executes its constituent statement (or block) repeatedly, testing expression before each iteration. As long as expression returns True, execution continues. Examples:

```
while Data[I] <> X do I := I + 1;
while I > 0 do
begin
  if Odd(I) then Z := Z * X;
  I := I div 2;
  X := Sqr(X);
end;
```

Repeat statements:
The syntax of a repeat statement is *repeat statement1; ...; statementn; until expression* where expression returns a Boolean value. The repeat statement executes its sequence of constituent statements continually, testing expression after each iteration. When expression returns True, the repeat statement terminates. The sequence is always executed at least once because expression is not evaluated until after the first iteration. Examples:

```
repeat
  K := I mod J;
  I := J;
  J := K;
until J = 0;
```

For statements:
Scripter support for statements with the following syntax: *for counter := initialValue to finalValue do statement* For statement set counter to initialValue, repeats execution of statement (or block) and increment value of counter until counter reachs finalValue.
Examples:

SCRIPT 1:
```
for c:=1 to 10 do
a:=a+c;
```

SCRIPT 2:
```
for i:=a to b do
begin
  j:=i^2;
  sum:=sum+j;
end;
```

Case statements:
Scripter support case statements with following syntax:
case selectorExpression of
caseexpr1: statement1;
...
caseexprn: statementn;
else
elsestatement;
end

If selectorExpression matches the result of one of caseexprn expressions, the respective statement (or block) will be execute. Otherwise, elsestatement will be execute. Else part of case statement is optional. Different from Delphi, case statement in script doesn't need to use only ordinal values. You can use expressions of any type in both selector expression and case expression. Example:

```
case uppercase(Fruit) of
  'lime': ShowMessage('green');
  'orange': ShowMessage('orange');
  'apple': ShowMessage('red');
else
  ShowMessage('black');
end;
```

Function and procedure declaration:
Declaration of functions and procedures are similar to Object Pascal in Delphi, with the difference you don’t specify variable types. Just like OP, to return function values, use implicited declared result variable. Parameters by reference can also be used, with the restriction mentioned: no need to specify variable types. Some examples:
```
procedure HelloWord;
begin
  ShowMessage('Hello world!');
end;
procedure UpcaseMessage(Msg);
begin
  ShowMessage(Uppercase(Msg));
end;
function TodayAsString;
begin
  result:=DateToStr(Date);
end;
function Max(A,B);
begin
  if A>B then
  result:=A
else
  result:=B;
end;
procedure SwapValues(var A, B);
Var Temp;
begin
  Temp:=A;
  A:=B;
  B:=Temp;
end;
```

Example of a Editbox where there is a time in minutes and you want it to display it in hours and minutes so 120 minutes will be displayed as 200.
The components where you can specify a device address has also the property Value. This property can be used in the script.

In de event : **OnGetValue**

```
var hours,minutes;
hours := TRUNC(Value/60);
minutes := Value - hours *60;

Value := hours*100 + minutes;
```

In de event: **OnSetValue**

```
var hours,minutes;
hours := TRUNC(Value/100);
minutes := Value - hours *100;

Value := hours*60 + minutes;
```