SIEMENS



# SIMOTION ProjectGenerator

Parameter Manual

03/2016

# Legal information

## Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

| ⚠DANGER |
| --- |
| indicates that death or severe personal injury **will** result if proper precautions are not taken. |

| ⚠WARNING |
| --- |
| indicates that death or severe personal injury **may** result if proper precautions are not taken. |

| ⚠CAUTION |
| --- |
| indicates that minor personal injury can result if proper precautions are not taken. |

| NOTICE |
| --- |
| indicates that property damage can result if proper precautions are not taken. |

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

## Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

## Proper use of Siemens products

Note the following:

| ⚠WARNING |
| --- |
| Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed. |

## Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

## Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 03/2016 Subject to change

# Preface

## General information

**Note**

The standard applications are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. The standard applications do not represent specific customer solutions, but are only intended to provide support for typical tasks. You are responsible for the proper operation of the described products. These standard applications do not relieve you of your responsibility regarding the safe handling when using, installing operating and maintaining the equipment. By using these standard applications, you agree that Siemens cannot be made liable for possible damage beyond the mentioned liability clause. We reserve the right to make changes and revisions to these standard applications at any time without prior notice. In the case of any differences between the suggestions made in these standard applications and other publications from Siemens, such as catalogs, the contents of the other documentation have priority.

## Warranty conditions, liability, and support

If the application has been made available free of charge, the following applies:

We do not provide a warranty for any of the information contained in this document.

All other rights and claims against Siemens AG irrespective of legal basis are excluded. In particular claims for damages against Siemens AG in the case of product outage, downtime, loss of profit, either directly, indirectly or consequential damage are excluded.

This does not apply when liability is compulsory by law, e. g. in the case of the Product Liability Act, premeditation, an act of gross negligence by superiors and managerial staff of Siemens AG or in cases of fraudulent concealment of defects.

This limitation of liability also applies to sub-contractors, suppliers, delegates, superiors and managerial staff of Siemens AG.

German law shall apply to this agreement for customers with head offices in Germany; Swiss law for customers with head offices outside Germany. Application of the United Nations Convention on Contracts for the International Sale of Goods as of 11.04.1980 (CISG) is excluded.

If the application has been made available against payment, the appropriate alternative applies for the respective business transaction:

- Alternative 1: (Internal business)

If nothing else has been negotiated, then the "Conditions for the supply and services in Siemens internal business" applies in the version that is valid at the time that the equipment is purchased.

- Alternative 2: (Domestic business of Siemens AG)

If nothing else was negotiated, the "General License Conditions for Software for Automation and Drives for Customers with a Registered Office in Germany" valid at the time of sale are applicable.

- Alternative 3: (Direct export business of Siemens AG)

If nothing else has been negotiated, then the "General License Conditions for Software Products for Automation and Drives for Customers with a Seat or Registered Office outside Germany", valid at the time of sale, are applicable.

It is not permitted to distribute or duplicate these application examples in any form including excerpts thereof without the express consent of Siemens Industry Sector.

## Notice regarding export identification codes

AL: N

ECCN: N

## About this document

## Objective

This manual describes the commands of the XML interface available with the ProjectGenerator. The ProjectGenerator is an addition to the SIMOTION motion control system.

The manual is an addition to the SIMOTION standard documentation.

**Note**

This document does not claim to contain all details on devices in any version or to take all conceivable operational cases and applications into account.

Should you require further information or encounter specific problems not covered in enough detail for your field of application, please contact your local Siemens office.

## Target group

This document is intended for programmers.

# Table of contents

# Templates and examples

# 1

## Templates for XML files

Detailed and executable templates for XML files are supplied with the ProjectGenerator and can be used as a copy template or as an orientation aid for your own expansions.

## Example

The following extract from the SIMOTION *StartupCheck.XML* standard module provides an overview of the interaction of the individual commands in an XML description file.

Table 1- 1 Extract from the XML description file of the *StartupCheck* standard module

```
<CommandList
    Name="pStartupCheck"
    DisplayText="Use Startup Check"
    MaxNumberOfModules="1"
    Mode="UnitOnly"
    IsPTemplNecessary="True"
    ModulInfoFile="SIMOTION\EquipmentModules\V4_3\StartupCheck\
                   SIMOTION_StartupCheck_V1_0.pdf">
    <Command ID="1" Name="ImportUnit" NCID="2">
        <Parameter
            Name="pStartupCheck"
            Path="SIMOTION\EquipmentModules\V4_3\StartupCheck\Data\Units\
                  pStartupCheck.xml"
            ForceImport="true" />
    </Command>
    <Command ID="2" Name="setValue" NCID="3">
        <Parameter
            Value="__CALL_MAXNUMBER_OF_AXES" Name="STARTUPCHECK_NUMBER_OF_AXES"
            TargetName="pStartupCheck"
            Type="Unit_Constant" />
        <Parameter Value="__CALL_MAXNUMBER_OF_EXTERNALENCODERS"
            Name="STARTUPCHECK_NUMBER_OF_EXTERNAL_ENCODERS"
            TargetName="pStartupCheck"
            Type="Unit_Constant" />
        <Parameter Value="__CALL_MAXNUMBER_OF_PERIPHERAL_DEVICES"
            Name="STARTUPCHECK_NUMBER_OF_PERIPHERAL_DEVICES"
            TargetName="pStartupCheck"
            Type="Unit_Constant" />
    </Command>
    <Command ID="3" Name="ChangeForm">
        <Control
            Action="add"
            Type="Button"
            Name="BT_Help"
            Text="Help"
            Location="12, 491"
            Size="130, 30"
            Enabled="true"
            Visible="true"
            ToolTip="Show help information">
                <Events>
                    <Click code="MyApp.NextCommand(2000)" />
                </Events>
        </Control>
```

Table 1- 2 Continuation of the extract from the XML description file of the *StartupCheck* standard module

```
<Control
    Action="add"
    Type="Button"
    Name="BT_Exit"
    Text="Exit"
    Location="12, 531"
    Size="130, 30"
    Enabled="true"
    Visible="true"
    ToolTip="Abort this program">
        <Events>
            <Click code="MyApp.NextCommand(0)" />
        </Events>
</Control>
    <Control Action="add"
        Type="Label"
        Name="LBL_Head_Info"
        Text="Startup Check - configuration"
        BackColor="__call_SetColor(Transparent)"
        AutoSize="true" Location="173, 110" />
 ...
```

If the example is completed further (as shown in the executable supplied file *StartupCheck.XML*), the ProjectGenerator screenshot for the example appears as follows:

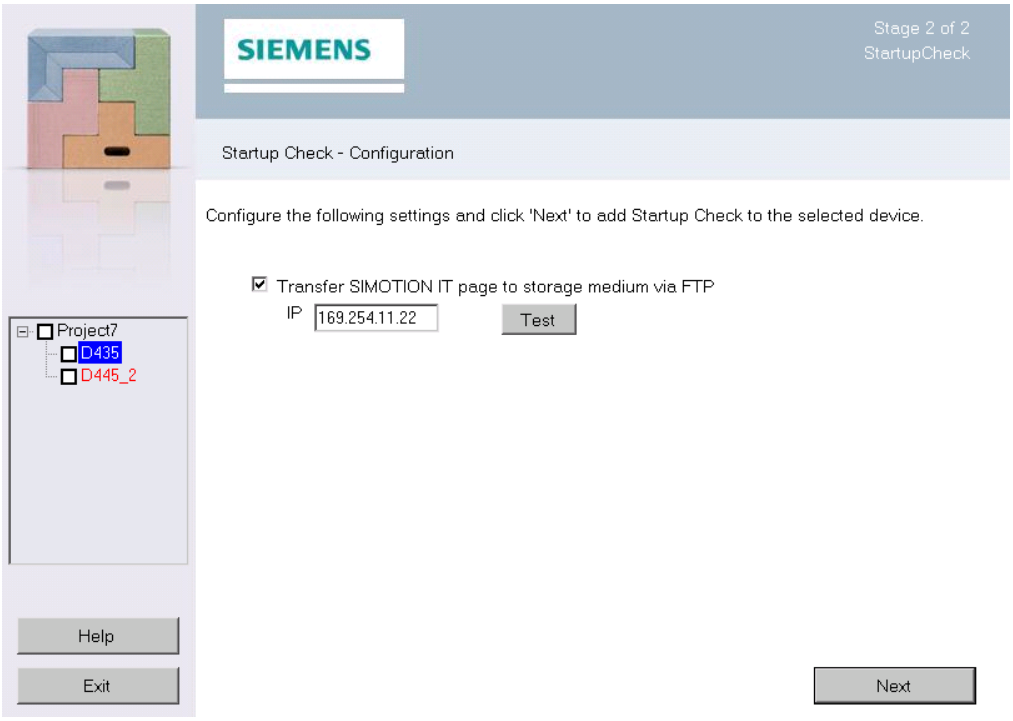



Image 1-1 ProjectGenerator screenshot

# Commands for creating a SIMOTION device 2

## 2.1 Overview of parameters

The table below provides an overview of the most important parameters that can be set for the commands in the ProjectGenerator. The parameters are listed together with their meaning, the data type, their possible values, and an example.

Table 2- 1 Overview of the commonly used parameters

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| Action | Specifies the action that is to be executed. | STRING | Add<br>Remove | `Action="add"` |
| Alignment | Specifies the alignment of the text within the control. | ENUM | Default<br>Left<br>Top<br>SnapToGrid | `Alignment="HorizontalAlignment.Left"` |
| AutoSize | Specifies whether the size of the element is to be determined automatically. | BOOL | True<br>False (default) | `AutoSize="true"` |
| BackColor | Specifies the background color. All colors that are defined in the .Net object *System.Drawing.Color* are possible here (e.g. White, Green, Blue, Red, LightSlateGray, etc.). A transfer of the RGB color codes is not possible. | Is defined via the *SetColor* command. | --- | `BackColor="__call_setcolor(Transparent)"` |
| Checked | Specifies whether the element is activated or not. | BOOL | True<br>False (default) | `Checked="true"` |
| DropDownStyle | Specifies the display type of the control. | ENUM | Simple<br>DropDown (default)<br>DropDownList | `DropDownStyle = "DropDownList"` |
| Enabled | Specifies whether an element can be operated. | BOOL | True (default)<br>False | `Enabled="true"` |
| FullRowSelect | Specifies whether the whole row is selected. | BOOL | True<br>False (default) | `FullRowSelect="true"` |
| HideSelection | Specifies whether the selection is hidden when the control loses focus. | BOOL | True<br>False (default) | `HideSelection="false"` |

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| Location | Specifies the position on the user interface (x, y). | INTEGER, INTEGER | 0.0 | `Location="450,200"` |
| MultiSelect | Specifies whether multiple row selection is permitted | BOOL | True<br>False (default) | `MultiSelect="false"` |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. | STRING | --- | `Name="LV_Projects"` |
| Size | Specifies the size of the control (width, height) if *AutoSize* is not selected. | INTEGER, INTEGER | 0.0 | `Size="150,13"` |
| SizeMode | Specifies the display of the picture. | ENUM | Normal (Default)<br>StretchImage<br>AutoSize<br>CenterImage<br>Zoom | `SizeMode = "StretchImage"` |
| Source | Transfers the content of the control element if this is generated via a system function. | STRING for function call | --- | `Source="__call_GetAvailableDeviceVersions"` |
| Text | Specifies the text which is to be displayed on the element. | STRING | --- | `Text="Button1"` |
| Tooltip | Specifies the text of the tooltip for the element. | STRING | --- | `ToolTip="Select the version of the new device"` |
| Type | Specifies the type of the control. | STRING | Button<br>Label<br>TextBox<br>ListBox<br>ListView<br>RadioButton<br>CheckBox<br>ComboBox<br>Picture | `Type="TextBox"` |
| View | Specifies the view of the control. | ENUM | LargeIcon (default)<br>Details<br>SmallIcon<br>List<br>Title | `View="Details"` |
| Visible | Specifies whether an element is visible. | BOOL | True (default)<br>False | |

Table 2- 2 Overview of the commonly used parameters for items

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| **Items** | | | | |
| Items | Accesses the source input via a reference (@source. …):<br>Select which elements from the system function call are to be selected. All elements from the *Source* parameter are then accepted. | STRING | --- | `items="@source.name"` |
| Name | Elements can be added directly here if no elements have been specified via the *Item* tag. | STRING | --- | |
| SelectedIndex | Specifies the preselection of the current index of the combo box. | INTEGER | 0 | `SelectedIndex ="0"` |

Table 2- 3 Overview of the commonly used parameters for events

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| **Events** | | | | |
| CheckedChanged | Is triggered when the state of the element has been changed. | --- | ---- | |
| Click | This action is triggered by clicking the button. | --- | --- | `<Click code="MyApp.NextCommand(0)" />` |
| SelectedIndexChanged | Is called when a different element has been selected within the control. | --- | --- | `SelectedIndexChanged code="@BT_Next@.Enabled=@LV_Projects@.SelectedItems.Count > 0"` |

All events provided by .NET can also be used here.

Table 2- 4 Overview of the commonly used parameters for limits

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| **Limits** | | | | |
| DataType | The data type that is to be entered in the text field can be specified here. Because of the default setting, certain characters are hidden and the value checked for validity. | ENUM | OBJECT<br>IPADDRESS<br>BOOL<br>BYTE<br>WORD<br>DWORD<br>USINT<br>SINT<br>UINT<br>INT<br>UDINT<br>DINT<br>REAL<br>LREAL<br>TIME<br>DATE<br>TOD<br>DT | `DataType="uint"` |
| Length | The length can also be limited here for the *Object* data type. | INTEGER | 255 | `Length="" Min="1" Max=""` |
| Max | A value can be limited to a maximum value here. | INTEGER | Dependent on the data type | |
| Min | A value can be limited to a minimum value here. | INTEGER | Dependent on the data type | |

Further information on the parameters can be found on the Internet in the Microsoft.NET Framework Class Library (http://msdn.microsoft.com/en-us/library/ff361664.aspx).

The following section contains the descriptions of all controls that can be inserted in the user interface.

# 2.2 General commands

## 2.2.1 ChangeForm

The most frequently required properties of the **controls** (**user interface elements**) are described in this section. It is possible to access all properties provided by the *Visual Basic .NET*.

A code, which is generated from the relevant code element of the XML tag during runtime, is transferred to the events of the user interface elements. This means that the code that is required for the execution is only compiled at the respective position and then inserted.

If a reference is required to an active element of the user interface, the element can be accessed via the assigned name. For this purpose however, the @ character must be inserted before and after the name so that the correct reference can be found.

The syntax for the code must comply with *Visual Basic .NET*. The insert sequence for access to elements within a *ChangeForm* command is not relevant (i.e. access to a control can be from the first element (see the example below) directly to the last control.

### Example

A checkbox is available on the user interface and the status *Checked* is to be set:

Table 2- 5 Code example

```
Code="@<Name des Controls>@.Checked = True"
```

### 2.2.1.1 Button


## Parameters

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Text | Specifies the text which is to be displayed on the element. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the element (width, height) if *AutoSize* is not selected. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| Tooltip | Specifies the text of the tooltip for the element. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| BackColor | Specifies the background color. |
| **Events** | |
| Click | This action is triggered by clicking the button. |

## Example

Table 2- 6 Button code example

```
<Control
    Action="add"
    Type="Button"
    Name="BT_Button1"
    Text="Button1"
    Location="12, 531"
    Size="130, 30"
    Enabled="true"
    Visible="true"
    ToolTip="This is Button1">
    <Events>
        <Click code="MyApp.NextCommand(0)" />
    </Events>
</Control>
```

### 2.2.1.2 CheckBox


## Parameters

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Text | Specifies the text which is to be displayed on the element. |
| BackColor | Specifies the background color. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| Location | Specifies the position on the user interface (x, y). |
| Checked | Specifies whether the checkbox is activated or not. |
| Size | Specifies the size of the control (width, height) if *AutoSize* is not selected. |
| Tooltip | Specifies the text of the tooltip for the element. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| **Events** | |
| CheckedChanged | Is triggered when the state of the checkbox has been changed. |

## Example

Table 2- 7 CheckBox code example

```
<Control
    Action="add"
    Type="CheckBox"
    Name="CB_DRIVE_OBJECT_diagnostics"
    Text="CheckBox1"
    BackColor="__call_SetColor(Transparent)"
    AutoSize="true"
    Location="200,277"
    Checked="true"
    Size="200,17"
    ToolTip="This is CheckBox1">
</Control>
```

### 2.2.1.3 ComboBox

Image 2-1 ComboBox

## Parameter

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Location | Specifies the position on the user interface (x, y). |
| DropDownStyle | Specifies the display type of the control. |
| Size | Specifies the size of the control (width, height) if *AutoSize* is not selected. |
| Source | Transfers the content of the control element if this is generated via a system function. |
| Tooltip | Specifies the text of the tooltip for the element. |
| BackColor | Specifies the background color. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| **Items** | |
| Name | Elements can be added directly here if no elements have been specified via the *Item* tag. |
| Alignment | Specifies the alignment of the text within the control. |
| SelectedIndex | Specifies the preselection of the current index of the combo box. |
| Items | Accesses the source input via a reference (@source. …):<br>Select which elements from the system function call are to be selected. All elements from the *Source* parameter are then accepted. |
| **Events** | |
| SelectedIndexChanged | Is called when a different element has been selected within the control. |

## Example

Table 2- 8 ComboBox code example

**Generating items via a system function**

```
<Control
    Action="add"
    Type="ComboBox"
    Name="CB_Name_New_Device_AvailableDeviceVersions"
    Location="550,300"
    DropDownStyle = "DropDownList"
    Size="150,13"
    Source="__call_GetAvailableDeviceVersions"
    ToolTip="Select the version of the new device"
    BackColor="__call_setcolor(Transparent)">
    <Items
        Name=""
        Alignment="HorizontalAlignment.Left"
        SelectedIndex ="0"
        Items="@source.name">
    </Items>
    <Events>
        <SelectedIndexChanged code="MyApp.NextCommand(54)"/>
    </Events>
</Control>
```

**Entering items manually**

```
<Control
    Action="add"
    Type="ComboBox"
    Name="CB_EncoderType"
    Location="620,590"
    Size="120, 15"
    Autosize="false"
    DropDownStyle="DropDownList"
    BorderStyle="none">
    <Items Name="Incremental"
        Size="100"
        Alignment="HorizontalAlignment.Left"
        items="">
    </Items>
    <Items Name="Absolute"
        Size="100"
        Alignment="HorizontalAlignment.Left"
        items="">
    </Items>
</Control>
```

### 2.2.1.4 Label


## Parameters

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Text | Specifies the text which is to be displayed on the element. |
| BackColor | Specifies the background color. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the element (width, height) if *AutoSize* is not selected. |
| Visible | Specifies whether an element is visible. |
| Tooltip | Specifies the text of the tooltip for the element. |

## Example

Table 2- 9 Label code example

```
<Control
    Action="add"
    Type="Label"
    Name="LBL_Label1"
    Text="Label1"
    BackColor="__call_SetColor(Transparent)"
    AutoSize="true"
    Location="173, 110" >
</Control>
```

### 2.2.1.5 ListBox

## Parameters

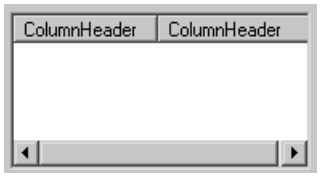
| Parameter | |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the button (width, height) if *AutoSize* is not selected. |
| Source | Transfers the content of the control element if this is generated via a system function. |
| Tooltip | Specifies the text of the tooltip for the element. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| BackColor | Specifies the background color. |
| Text | Specifies the text which is to be displayed on the element. |
| **Events** | |
| SelectedIndexChanged | Is called when a different element has been selected within the control. |
| **Items** | |
| items | Accesses the source input via a reference (@source. …):<br>Select which elements from the system function call are to be selected. All elements from the *Source* parameter are then accepted. |

## Example

Table 2- 10 ListBox code example

```
<Control
    Action="add"
    Type="ListBox"
    Name="LB_Axes"
    Location="340,250"
    Size="150,180"
    Source="__call_GetAvailableAxesByType()"
    ToolTip="Show all axes of the selected device">
    <Items items="@source.name"/>
</Control>
```

### 2.2.1.6 ListView




## Parameter

| Parameter | Meaning |
|---|---|
| **Parameter** | **Meaning** |
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| View | Specifies the view of the control. |
| MultiSelect | Specifies whether multiple row selection is permitted. |
| FullRowSelect | Specifies whether the whole row is selected. |
| HideSelection | Specifies whether the selection is hidden when the control loses focus. |
| Text | Specifies the text which is to be displayed on the element. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the control (width, height) if *AutoSize* is not selected. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| Tooltip | Specifies the text of the Tooltips for the element. |
| Source | Transfers the content of the control element if this is generated via a system function. |
| BackColor | Specifies the background color. |
| **Events** | |
| Selected IndexChanged | Is called when a different element has been selected within the control. |
| **ListViewItem** | |
| Name | Specifies the name of the main element. |
| Size | Specifies the size of the display within the control (can be visible or not depending on the selected view of the control). |
| Alignment | Alignment of the text within the control. |
| **ListViewSubItem** | *ListViewSubItem* can be used several times below the *ListViewItems*. Each *ListViewSubItem* defines a new column in the view. |
| Name | Specifies the name and the text that is entered as header in the table. |

| | |
|---|---|
| Size | Specifies the size of the display within the column (can be visible or not depending on the selected view of the control). |
| Alignment | Specifies the alignment of the text within the column. |
| Item | Accesses the source input via a reference (@source. …):<br>Select which elements from the system function call are to be selected. All elements from the *Source* parameter are then accepted. |

## Example

Table 2- 11 ListView code example

```
<Control
 Action="add"
 Type="ListView"
 View="Details"
 Name="LV_Projects"
 MultiSelect="false"
 FullRowSelect="true"
 HideSelection="false"
 Location="170, 220"
 Size="600, 300"
 Source="__call_GetProjects">
 <ListViewItem
 Name="Project"
 Size="100"
 Alignment="HorizontalAlignment.Left"
 Item="@source.name" >
 <ListViewSubItem
 Name="Projectpath"
 Size="300"
 Alignment="HorizontalAlignment.Left"
 Item="@source.logpath" />
 <ListViewSubItem
 Name="Last modified"
 Size="130"
 Alignment="HorizontalAlignment.Left"
 Item="@source.modified" />
 <ListViewSubItem
 Name="Creator"
 Size="50"
 Alignment="HorizontalAlignment.Left"
 Item="@source.creator" />
 </ListViewItem>
 <Events>
 <SelectedIndexChanged
 code="@BT_Next@.Enabled=@LV_Projects@.SelectedItems.Count > 0"/>
 </Events>
</Control>
```

### 2.2.1.7 Picture

## Parameter

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the picture (width, height) if *AutoSize* is not selected. |
| SizeMode | Specifies the display of the picture. |
| ImageLocation | Specifies the path to the picture. The path can also be specified relatively. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| Tooltip | Specifies the text of the tooltip for the element. |

## Example

Table 2- 12 Picture code example

```
<Control
    Action="add"
    Type="Picture"
    Name="PB_Example2"
    Location="350, 0"
    Size="250,150"
    SizeMode = "StretchImage"
    ImageLocation ="Pictures\Example2.png">
</Control>
```

### 2.2.1.8 RadioButton


## Parameters

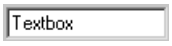
| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| BackColor | Specifies the background color. |
| Text | Specifies the text which is to be displayed on the element. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| Enabled | Specifies whether an element can be operated. |
| Location | Specifies the position on the user interface (x, y). |
| Checked | Specifies whether the radio button is activated or not. |
| Size | Specifies the size of the button (width, height) if *AutoSize* is not selected. |
| Tooltip | Specifies the text of the tooltip for the element. |
| Visible | Specifies whether an element is visible. |

## Example

Table 2- 13 RadioButton code example

```
<Control
    Action="add"
    Type="RadioButton"
    Name="RB_RadioButton1"
    BackColor="__call_SetColor(Transparent)"
    Text="RadioButton1"
    AutoSize="true"
    Enabled="true"
    Location="170,216"
    Checked="true"
    Size="200,17"
    ToolTip="This is RadioButton1">
</Control>
```

### 2.2.1.9 TextBox




## Parameter

| Parameter | Meaning |
|---|---|
| Action | Specifies the action that is to be executed. |
| Type | Specifies the type of the control. |
| Name | Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code. |
| Text | Specifies the text which is to be displayed on the element. |
| Location | Specifies the position on the user interface (x, y). |
| Size | Specifies the size of the element (width, height) if *AutoSize* is not selected. |
| Enabled | Specifies whether an element can be operated. |
| Visible | Specifies whether an element is visible. |
| AutoSize | Specifies whether the size of the element is to be determined automatically. |
| BackColor | Specifies the background color. |
| **Limits** | |
| DataType | The data type that is to be entered in the text field can be specified here. Because of the default setting, certain characters are hidden and the value checked for validity. |
| Length | The length can also be limited here for the *Object* data type. |
| Min | A value can be limited to a minimum value here. |
| Max | A value can be limited to a maximum value here. |

## Data types for the Limit tag data type

| DataType | |
|---|---|
| Object | Checks the syntax according to SIMOTION SCOUT object specifications. |
| IPAddress | Checks for length, value and SCOUT syntax in the IPv4 aaa.bbb.ccc.ddd format. |
| Bool | Checks for value and SCOUT syntax. |
| Byte | Checks for length, value and SCOUT syntax. |
| Word | Checks for length, value and SCOUT syntax. |
| DWord | Checks for length, value and SCOUT syntax. |
| USint | Checks for length, value and SCOUT syntax. |
| Sint | Checks for length, value and SCOUT syntax. |
| Uint | Checks for length, value and SCOUT syntax. |
| Int | Checks for length, value and SCOUT syntax. |
| Udint | Checks for length, value and SCOUT syntax. |
| Dint | Checks for length, value and SCOUT syntax. |
| Real | Checks for length, value and SCOUT syntax. |
| LReal | Checks for length, value and SCOUT syntax. |
| Time | Checks for length, value and SCOUT syntax. |
| Date | Checks for length, value and SCOUT syntax. |
| Tod | Checks for length, value and SCOUT syntax. |
| DT | Checks for length, value and SCOUT syntax. |
| Project | Specifies the name of a new project in framework.xml. |

## Example

Table 2- 14 TextBox code example

```
<Control
    Action="add"
    Type="TextBox"
    Name=" TB_Size_Of_Active_Messages_List"
    Text="__call_GetValueOfLibVariable
         (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES,LMsgHdl, cPublic, 100)"
    Location="450,200"
    Size="50,20"
    ToolTip="Size of active messages list">
    <Limits DataType="uint" Length="" Min="1" Max="" />
</Control>
```

### 2.2.2 ActivateLogging

The logging functionality of the project generator activities is activated with this command.

#### Command parameters

This command does not have any parameters.

#### Example

Table 2- 15 Code example

```
<Command
    ID="100"
    Name="ActivateLogging"
    NCID="101">
</Command>
```

### 2.2.3 BrowseProject

A file browser is opened with this command in order to select a STEP 7 project. When a valid project is selected, it is opened.

#### Command parameters

This command does not have any parameters.

#### Example

Table 2- 16 Code example

```
<Click code="If MyApp.BrowseProject Then MyApp.NextCommand(32)"/>
```

## 2.2.4 CheckPath

This command can be called in the code of an event with the *MyApp.* prefix.

The current value is read from the transferred control with this command. A check is performed as to whether the path is available.

### Command parameters

| Parameter | |
|---|---|
| Control | Transfer of the control with the start and target path |

### Example

Table 2- 17 Code example

```
<Leave code="MyApp.CheckPath(@TB_Path@)"/>
```

## 2.2.5 CheckProjectPath

This command can be called in the code of an event with the *MyApp.* prefix.

This command checks whether the project path already exists.

### Command parameters

| Transfer parameters | |
|---|---|
| Path | Path of the project |
| Name | Name of the project |

### Example

Table 2- 18 Code example

```
<Click code="If MyApp.CheckProjectPath(@TB_Path@.Text,@TB_Name@.Text)
             Then MyApp.NextCommand(26)"/>
```

## 2.2.6 DestroyForm

All control elements of the user interface are deleted with this command. Exceptions are the device overview and the display of the current configuration step.

This command is used when the configuration of a standard module stretches over several pages.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 19 Code example

```
<Command
    ID="1"
    Name="DestroyForm"
    NCID="20">
</Command>
```

## 2.2.7 FolderBrowser

This command can be called in the code of an event with the *MyApp.* prefix.

The current value from the transferred control is read out with this command and the browser window with the read value started. After a path has been selected, the command writes the path back to the control. The return value of the function is a Boolean value which indicates whether the selection has been confirmed with OK.

### Command parameters

| Parameter | |
|---|---|
| Control | Transfer of the control with the start and target path |

### Example

Table 2- 20 Code example

```
<Click code="MyApp.FolderBrowser(@TB_Path@)"/>
```

## 2.2.8 NextCommand

This command is only called in the code of an event with the *MyApp.* prefix.

A jump is performed to the *Element* command line with the transferred ID with this command. If the ID equal to 0 is transferred, the Exit dialog box is opened.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| NCID | Transfer of the next command ID |

### Example

Table 2- 21 Code example

```
<Click code="MyApp.NextCommand(0)"/>
```

## 2.2.9 OpenFile

A file with the standard program set in Windows is started with this command. If several files are to be opened, the parameter line can be be executed several times.

### Command parameters

**Parameter**

| | |
|---|---|
| Path | Specifies the path to the file |

### Example

Table 2- 22 Code example

```
<Command
    ID="100"
    Name="OpenFile"
    NCID="20">
    <Parameter

Path="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\doc.pdf"/>
</Command>
```

## 2.2.10 PingToIPAddress

A ping to an IP address is performed with this command and the result is output in a message box. If several pings are to be performed, the parameter line can be called several times.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Value | Target IP address |

### Example

Table 2- 23 Code example

```
<Command
    ID="100"
    Name="PingToIPAddress"
    NCID="20">
    <Parameter Name="Parameter1" value="__ref_TB_IPAddress.Text">
    </Parameter>
</Command>
```

## 2.2.11 ReadNextEquipmentModuleConfig

This command informs the ProjectGenerator that the configuration of an equipment module has been completed. This command is therefore the last command that is executed for the configuration of an equipment module. If a further command has been selected in the selection for the equipment module which follows the current command, the configuration is continued with the next equipment module. If this is not the case, the "Project Generation" page is opened after the call of this command.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 24 Code example

```
<Command
    ID="100"
    Name="ReadNextEquipmentModuleConfig"/>
```

## 2.2.12 SetColor

This command defines colors, e.g the color of the background.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Color | All colors that are defined in the .Net object System.Drawing.Color are possible here (e.g. White, Green, Blue, Red, LightSlateGray, etc.).<br>A transfer of the RGB color codes is not possible. |

### Example

Table 2- 25 Code example

```
BackColor="__call_SetColor(Transparent)"
```

## 2.2.13 SetFileSystemTransfer

The source files (LocalPath) are copied to the target path with this command.

### Command parameters

**Parameter**

| | |
|---|---|
| LocalPath | Specifies the local path |
| Path | Specifies the target path |

With the path specification, the ProjectGenerator also replaces entries such as `WindowsTemp` and `%Temp%` instead of `C:\Temp`.

### Example

Table 2- 26 Code example

```
<Command
    ID="100"
    Name="SetFileSystemTransfer"
    NCID="20">
    <Parameter
        LocalPath="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\USER_DIR"
        Path ="WindowsTemp\USER\SIMOTION\USER_DIR\"/>
</Command>
```

## 2.2.14 SetFTPTransfer

This command initiates an FTP file transfer that transfers the entire contents of the source path (LocalPath) to the target path (ServerPath) on the FTP server with the specified IP address. Whereby already existing files are overwritten, additional existing files are not deleted.

The parameter line can be called several times.

### Command parameters

| Parameter | |
|---|---|
| IPAddress | Target IP address |
| LocalPath | Specifies the local path |
| ServerPath | Specifies the target path |

### Example

Table 2- 27 Code example

```
<Command
    ID="100"
    Name="SetFTPTransfer"
    NCID="20">
    <Parameter
        IPAddress="TB_IPAddress"
        Lo-
calPath="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\HMI"
        ServerPath="/USER/SIMOTION/HMI/FILES"/>
</Command>
```

## 2.2.15 ShowAboutBox

The About box is activated with this command. The About box contains information on the precise product designation, the installed version, the company name, and the copyright.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 28 Code example

```
<Command
    ID="6"
    Name="ShowAboutBox"/>
```

## 2.2.16 Step7DefaultPath

This command can be transferred directly to a control property via the *__Call_* prefix.

The current default path set for the projects in STEP 7 is returned with this command.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 29 Code example

```
Text="__call_Step7DefaultPath"
```

## 2.2.17 ExecuteScript

With this command, a user script is executed after generation. The script may be at the project, device or TO/DO level.

Project level:

- Specify the name of the script

Device level:

- Specify the name of the script
- Specify the device name

TO/DO level:

- Specify the name of the script
- Specify the device name
- Specify the TO/DO name

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the script that is to be executed. |
| Device | Optional: The name of the device on which the script is located. |
| Destination | Optional: The name of the TO/DOs on which the script is located |

### Example

Table 2- 30 Code example

```
<Command
      ID="12"
      Name="ExecuteScript">
      <Parameter
            Name="ConnectDOs"
            Device="SINAMICS_Integrated"
            Destination="DO_1" />
</Command>
```

## 2.2.18 ReadDirectoryFilenames

All file names in a path are returned with this command. The files can be preselected with a filter. The function returns an object of the *System.Collections.Arraylist* type. This object contains a collection of *System.Collections.Generic.SortedDictionary* objects which correspond to the files in the folder. The entries in a SortedDictionary object can be accessed with keys. This command can be transferred directly to a control property via the *__Call_* prefix. In this case, the names of the files are returned.

A *SortedDictionary* object is returned with the following keys for each file in the specified folder. When using keys in the source code, please note that they must be written in capital letters.

| Key | Value |
|---|---|
| EXTENSION | File ending including period (e.g. .txt) |
| FULLNAME | Complete path of the file including file name and ending. |
| NAME | Name of the file without ending |

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Filter | Optional: Specification of a filter for file extensions, e.g., ".xml". If a filter is not specified, all files are returned |
| StartDirectory | Path from which the file names should be returned. |

### Code example

**Transfer to a control**

```
<Control Action="add"
         Type = "ComboBox"
         Name = "CmB_FileNames"
         DropDownStyle = "DropDownList"
         Location = "340, 200"
         Size = "100, 20"
  Source =

"__call_readdirectoryfilenames('','SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\
Data\Units\')"
  Autosize ="False">
  <Items Name=""
              Size="100"
              Alignment="HorizontalAlignment.Left"
              items="@Source.Name">
      </Items>
</Control>
```

**Use in the source code**

```
REM Path of the directory with the files
```

**Use in the source code**

```
Dim FilesDirectory As String =
'SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\Data\Units\'

REM Read the data into the arraylist FileList
Dim FileList As System.Collections.Arraylist = MyApp.myIsl.ReadDirectoryFilenames('XML',
FilesDirectory)

REM MessageBox with the number of files in the directory
MsgBox (FileList.Count)

Dim OutputTxt As String
REM Loop through the SortedDictionary Objects in FileList
For Each tmpXMLFile As System.Collections.Generic.SortedDictionary(Of String, String) in
FileList

     REM Output the Name of the File
     MsgBox(tmpXMLFile('NAME'))
     IF tmpXMLFile('EXTENSION').ToLower = '.txt' Then
             REM do something if it’s a text file
     End IF

     REM String to collect all keys and the corresponding values of a file
     OutputTxt = ''

     REM Loop through the KeyValuePair Objects in every file
     For Each keyValPair As KeyValuePair(Of String, String) In tmpXMLFile
          REM Add key and value to the output text
          OutputTxt += keyValPair.Key + '       ' + keyValPair.Value + vbcrlf
     Next
     REM MessageBox with keys and values of a file
     MsgBox(OutputTxt)
Next
```

## 2.2.19 ReadDirectoryNames

All directory names in a path are returned with this command. The function returns an object of the *System.Collections.Arraylist* type. This object contains a collection of *System.Collections.Generic.SortedDictionary* objects which correspond to the directories in the folder. The entries in a *SortedDictionary* object can be accessed with keys. This command can be transferred directly to a control property via the *__Call_* prefix. In this case, the names of the files are returned.

A *SortedDictionary* object is returned with the following keys for each directory in the specified folder. When using keys in the source code, please note that they must be written in capital letters.

| Key | Value |
|---|---|
| FULLNAME | Complete path of the file including file name and ending. |
| NAME | Name of the file without ending. |

### Command parameters

**Transfer parameters**

| | |
|---|---|
| **StartDirectory** | Path from which the directory names should be returned. |

### Example

Table 2- 31 Code example

**Transfer to a control**

```
<Control Action="add"
         Type = "ComboBox"
         Name = "CmB_Directories"
         DropDownStyle = "DropDownList"
         Location = "340, 200"
         Size = "100, 20"
          Source =

"__call_readdirectorynames('SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\Data')"
        <Items Name=""
              Size="100"
              Alignment="HorizontalAlignment.Left"
              items="@Source.Name">
      </Items>
</Control>
```

**Use in the source code**

```
REM Path of the directory
Dim DirectoryPath As String =
'SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\Data'
```

**Use in the source code**

```
REM Read the data into the arraylist Directory
Dim DirectoryList As System.Collections.Arraylist = My-
App.myIsl.ReadDirectoryNames(DirectoryPath)

REM MessageBox with the number of directories in the directory
MsgBox (DirectoryList.Count)

REM Loop through the SortedDictionary Objects in DirectoryList
For Each tmpXMLDirectory As System.Collections.Generic.SortedDictionary(Of String, String)
in DirectoryList

     REM Output the name of the directory
     MsgBox(tmpXMLDirectory ('NAME'))
     REM Output the fullname of the directory
     MsgBox(tmpXMLDirectory ('FULLNAME'))
Next
```

## 2.2.20 ReadXmlFile

With this command, an XML file can be read out generically and the user interface can be provided to fill several combo boxes with the same content without parameterizing it for each combo box, for example. This command can be transferred directly to a control property via the *__Call_* prefix. The return value of the function is a collection (Arraylist) of *SortedDictionary* objects which represent the XML elements. The keys correspond to the XML attributes. For the file in the table below, the keys are: NAME, TYPE, MOTIONTYPE, and VIRTUAL

### Example configuration of an XML file

Table 2- 32 Code example

```
<SimotionAxes>
<Axis Name="TO_Axis1" Type="Gear" MotionType="Rotatory" Virtual="True"  />
<Axis Name="TO_Axis2" Type="Pos" MotionType="Linear" Virtual="False"  />
</SimotionAxes>
```

**Note**

Name of the start tag and quantity and name of the attributes are generally freely selectable. Any number of XML elements can be programmed. If the command is transferred to a control property, there **must** be a 'Name' attribute. When accessing from the source code, this restriction does not apply.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| FilePath | Specification of the path to the XML file that should be read. |

### Example

Table 2- 33 Code example

**Transfer to a control**

```
<Control Action="add"
         Type = "ComboBox"
         Name = "CmB_AxesNames"
         DropDownStyle = "DropDownList"
         Location = "340, 200"
         Size = "100, 20"
         Source =
         "__call_readXmlFile('SIMOTION\EquipmentModules\V4_3 \
         DocExample\Data\SimotionAxes.xml')"
```

**Transfer to a control**

```
         Autosize ="False">
         <Items Name=""
              Size="100"
              Alignment="HorizontalAlignment.Left"
              items="@Source.Name">
      </Items>
 </Control>
```

**Use in the source code**

```
REM Path of the xml file
Dim XmlFilePath As String =
'SIMOTION\EquipmentModules\V4_3\DocExample\Data\SimotionAxes.xml'

REM Read the data in the xml File into the arraylist AxesList
Dim AxesList As System.Collections.Arraylist = MyApp.myIsl.ReadXmlFile(XmlFilePath)

REM MessageBox with the number of Items in the xml file
MsgBox (AxesList.Count)

Dim counter As Integer
Dim tmpAxis As System.Collections.Generic.SortedDictionary(Of String, String)
REM Loop through the SortedDictionary objects in AxesList
For counter = 0 To AxesList.Count -1

     tmpAxis = AxesList(counter)
     REM Output the name of the file
     MsgBox(tmpAxis ('NAME'))

     REM or
     REM MsgBox(AxesList(counter)('NAME'))

     IF tmpAxis('VIRTUAL')= True Then
        REM Do something if the axis is virtual
     End IF

Next
```

## 2.2.21 SetSubsystemProperty

Any permissible HW Config property can be set on a subsystem with this command.

---

**Note**

**There is no check by the ProjectGenerator.**
**Whether parameters are to be set and in which sequence depends solely on the user and the HW Config.**

---

### Command parameters

| Parameter | |
|---|---|
| CPUInterface | Specification of the active device interface on which the subsystem searched for is located. |
| Name | Specification of the parameter name from the HW Config. |
| Value | Specification of the value to be set. |

### Example

Table 2- 34 Code example

```
<Command
      ID="12"
      Name="SetSubsystemProperty">
      <Parameter
            Name=" "
            CPUInterface="PN1"
            Value="False"
            />
</Command>
```

## 2.2.22 SetDeviceProperty

The value of every permissible HW Config property (attribute) of a device can be set with this command. If it is the property of a module or a submodule, then the appropriate module (and submodule) number must be specified. It may be that the required property is within a field, as with *LocalInAddresses* or *LocalOutAddresses*. It is possible to optionally specify the name of the relevant object with *ObjectType*.

### Command parameters

| Parameter | |
|---|---|
| DeviceName | Specification of the sought device name. |
| Name | Name of the property/attribute. |
| Value | The value that is to be assigned. |
| SlotNumber | Optional: Specification of the slot number for the addressing of subordinate slots. |
| SubSlotNumber | Optional: Specification of the subslot number for the addressing of subordinate subslots. |
| ObjectType | Optional: Specification of the object to be addressed, e.g. *LocalInAddresses* or *LocalOutAddresses*. |

### Example

Table 2- 35 Code example

```
<Command ID="301" Name="SetDeviceProperty">
 <Parameter
 DeviceName="MyET200"
 CPUInterface="PN1"
 SlotNumber="3"
 SubSlotNumber=""
 Name="LocalAddress"
 Value="1500"
 ObjectType="LocalOutAddresses"
 />
</Command>
```

## 2.2.23 SetCompileOption

With this command, the compiler options can be set to projects, units, and units in libraries.

Any number of compiler options can be set with one call of the command. The parameter line must be filled in several times in this case.

To set compiler options to different levels in the project, the following parameters have to be set as follows:

- Project:
  - Library & Unit empty
- Unit:
  - Library leer
  - Unit Name
- Library:
  - Library Name
  - Unit Name

### Command parameters

| Parameter | |
|---|---|
| Library | Optional: Name of the library. |
| Unit | Optional: Name of the unit/name of the unit in the library. |
| CompileOption | Specifies the compiler option that should be set. |

### Example

Table 2- 36 Code example

```
<Command
      ID="10014"
      Name="setCompileOption"
      NCID="10015">
      <Parameter
            Library=""
            Unit=""
            CompileOption="-C lang_ext"/>
</Command>
```

## 2.2.24 DeleteCompileOption

The compiler options on an existing source (unit or library unit) or on the project can be deleted with this command.

Several sources can be addressed with one system call if the parameter line is integrated several times.

Several compiler options of a source can be deleted if the subparameter *CompileOption* below the parameter line is called several times.

### Command parameters

**Parameter**

| | |
|---|---|
| Library | Optional: Name of the target library. If no library unit is used, this parameter is omitted. |
| Unit | Optional: Name of the unit. If the project level is addressed, this parameter is omitted. |

**Subparameter**

**CompileOption**

| | |
|---|---|
| Name | Name of the compiler option to be deleted. |

### Example

Table 2- 37 Code example

```
<Command
      ID="100"
      Name="deleteCompileOption"
      NCID="10014">
      <Parameter
            Library=""
            Unit="">
            <CompileOption
            Name="-C lang_iec"/>
      </Parameter>
</Command>
```

## 2.2.25 SetTemporaryVariable

With this command, values in the data management of the ProjectGenerator can be stored temporarily.

Any number of values can be stored temporarily with one call of the command. The parameter line must be filled in several times in this case. The values can be read with the GetValueOfTemporaryVariable (Page 93) function.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the temporary variables. |
| Type | Optional: You can choose between the options LOCAL and GLOBAL.<br>Local temporary variables are automatically deleted when the next module is called; global variables are retained. If this parameter is not supplied, LOCAL is taken automatically. |
| Value | Value of the temporary variables. |

### Example

Two variables are saved. The first variable is called „Hello“ and has the value „World“. The second variable is called „ModuloLength“ and has the value "360".

Table 2- 38 Code example

**Use in a command**

```
<Command  ID="100011" Name="setTemporaryVariable"  NCID="">
      <Parameter  Name="Hello" Type ="Global"  Value="World"/>
      <Parameter  Name="ModuloLength"  Value="360"/>
</Command>
```

**Use in the source code**

```
MyApp.myISL.setTemporaryVariable('ProjectName' ,'Global', 'MyProject')
MyApp.myISL.setTemporaryVariable('ModuloLength' ,'Local', 360.0)
```

## 2.2.26 SetUnitInterfaceConnection

With this command, interface connections (USES and USELIB) from units to other units and libraries can be created.

Any number of connections can be created with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the target unit/library. |
| UnitName | Name of the unit to which the interface should be adjusted. |
| Type | USELIB; USES |
| Interface | IMPLEMENTATION; INTERFACE |

### Example

Table 2- 39 Code example

```
<Command
      ID="100011"
      Name="setUnitInterfaceConnection"
      NCID="">
      <Parameter
            Name ="dGlobal"
            UnitName ="pLcom"
            Type ="USES"
            Interface ="INTERFACE"/>
</Command>
```

## 2.2.27 SetIPAddress

With this command, the IP address of the active device can be set for the IE1 and IE2 interfaces.

Any number of properties of the tasks can be changed with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| IP1 | Part 1 of the IP address to be set. |
| IP2 | Part 2 of the IP address to be set. |
| IP3 | Part 3 of the IP address to be set. |
| IP4 | Part 4 of the IP address to be set. |
| SubNet1 | Part 1 of the subnet mask to be set. |
| SubNet2 | Part 2 of the subnet mask to be set. |
| SubNet3 | Part 3 of the subnet mask to be set. |
| SubNet4 | Part 4 of the subnet mask to be set. |
| RouterIP1 | Part 1 of the router IP address to be set |
| RouterIP2 | Part 2 of the router IP address to be set. |
| RouterIP3 | Part 3 of the router IP address to be set |
| RouterIP4 | Part 4 of the router IP address to be set. |
| Port | Specification of the port that should be changed (IE1 or IE2). |
| RouterActive | Specification of whether a router is used. |

### Example

Table 2- 40 Code example

```
<Command ID="100011"
      Name="SetIPAddress"
      NCID="100013">
      <Parameter
            IP1="192"
            IP2="168"
            IP3="0"
            IP4="1"
            SubNet1="255"
            SubNet2="255"
            SubNet3="255"
            SubNet4="0"
            RouterIP1="192"
            RouterIP2="168"
            RouterIP3="0"
            RouterIP4="1"
```

```
            Port="IE1"
            RouterActive="False"/>
</Command>
```

## 2.2.28 SetTaskConfig

With this command, properties of the tasks can be changed or set. Any number of properties of the tasks can be changed with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| TargetTask | Name of the target task. ALL can also be specified for the error response. All active tasks are processed in this regard. |
| Property | Name of the property to be set. |
| Value | Value of the property to be set. |

| Property | Possible values |
|---|---|
| Activate | TRUE, FALSE |
| Active | TRUE, FALSE |
| OverflowCount | 0-5 |
| ProgrammErrorMode | „CPU_STOP“ |
| | „FAULT_TASK“ |
| StackSize | Integer |
| StartupCondition | String |
| StartupDelay | Integer |
| TimeAllocation | Integer |
| TimeLevel | Integer |
| TimeMonitoring | „MEASURE_SYNC_FUNCS“ |
| | „EXCLUDE_SYNC_FUNCS“ |
| | „BLOCK_SYNC_FUNCS“ |
| TimeOutErrorMode | „CPU_STOP“ |
| | „FAULT_TASK“ |
| TimeRatio | Integer |

**Note**

The properties are not relevant to all tasks. For example, the Activate property can only be set for a MotionTask. You can find a detailed description of the parameters in the documentation on the SIMOTION SCOUT scripting interface.

### Example

Table 2- 41 Code example

```
<Command ID="210"
      Name="SetTaskConfig"
      NCID="1000">
      <Parameter
            TargetTask="All"
            Property="ProgramErrorMode"
            Value="FAULT_TASK"/>
      <Parameter
            TargetTask="Backgroundtask"
            Property="TimeOutErrorMode"
            Value="FAULT_TASK"/>
</Command>
```

## 2.2.29 StartupPath

This command returns the path to the main directory of the ProjectGenerator as a string. If the command is called via the VB code, this must be written as follows: *MyApp.StartupPath*

### Example

Table 2- 42 Code example

```
Use in the source code
REM Path of the ProjectGenerator
Dim ProjectGeneratorPath As String = MyApp.StartupPath
```

## 2.2.30 ExecuteCode

With this command, VB.NET code can be inserted in an equipment module which is integrated in the execution sequence of the individual commands. It is therefore possible, for example, to already execute the code specified in the command before the ChangeForm function is called. In this way, it is possible to determine information in advance and therefore to be able to influence the structure of the equipment module page accordingly.

It is also possible to execute this command several times with the MyApp.NextCommand, whereby the code can be used as function.

### Command parameters

| Parameter | |
|---|---|
| FunctionName | Function name. This must be unique within an equipment module. |
| Code | The VB.NET code of the function. |

### Example

Table 2- 43 Code example

```
<Command ID="1" Name="ExecuteCode" >
     <Parameter
          <FunctionName="SelectOptions"
          Code=" If System.IO.File.Exists('D:\SinamicsConfig.xml') Then
                    MyApp.NextCommand(100)
               Else
                    MyApp.NextCommand(101)
               End If"
     />
</Command>
```

## 2.2.31 UseSymbolicAssignment

The automatic activation of the symbolic assignment by the ProjectGenerator within the project can be influenced with this command. If this command is not used, the symbolic assignment is activated per default at the end of the project generation.

### Command parameters

| Parameter | |
|---|---|
| UseSymbolicAssign-ment | Specifies whether the symbolic assignment is to be activated or not. The values True and False are permitted. |

### Example

Table 2- 44 Code example

```
<Command ID="11" Name="UseSymbolicAssignment" NCID="20">
     <Parameter UseSymbolicAssignment="False" />
</Command>
```

## 2.2.32 WriteX142Channel

The inputs and outputs of the X142 interface on a SIMOTION D4x5-2 can be configured with this command. This function can only be used with SIMOTION SCOUT as of Version 4.4.

### Command parameters

| Parameter | |
|---|---|
| Channel | Number of the input or output. |
| Type | Name of the setting for this channel. |
| Value | Value of the setting for this channel. |

### Example

Table 2- 45 Code example

```
<Command ID="120" Name="WriteX142Channel">
     <Parameter Channel="1" Type="Inverter" Value="True"/>
     <Parameter Channel="2" Type="Function" Value="Output cam"/>
     <Parameter Channel="3" Type="Function" Value="Measuring input"/>
     <Parameter Channel="4" Type="Function" Value="DO"/>
     <Parameter Channel="5" Type="DIFilterTime" Value="1"/>
</Command>
```

## 2.2.33 SetInputAddressOfX142

The input address of the X142 interface on a SIMOTION D4x5-2 can be configured with this command. The function refers to the active SIMOTION device.

This function can only be used with SIMOTION SCOUT as of Version 4.4.

### Command parameters

| Parameter | |
|---|---|
| Address | Input address. |
| Device | Optional: Name of the SIMOTION device. If this parameter is not specified, the command is used on the active device. |

### Example

Table 2- 46 Code example

```
<Command ID="1104" Name="SetInputAddressOfX142" >
     <Parameter
          Address="112"
     />
</Command>
```

### 2.2.34 SetOutputAddressOfX142

The output address of the X142 interface on a SIMOTION D4x5-2 can be configured with this command. The function refers to the active SIMOTION device.

This function can only be used with SIMOTION SCOUT as of Version 4.4.

#### Command parameters

| Parameter | |
|---|---|
| Address | Output address. |
| Device | Optional: Name of the SIMOTION device. If this parameter is not specified, the command is used on the active device. |

#### Example

Table 2- 47 Code example

```
<Command ID="1105" Name="SetOutputAddressOfX142" >
     <Parameter
          Address="112"
     />
</Command>
```

## 2.3 Commands for inserting and deleting objects

### 2.3.1 CreateObject

An object that is defined via *Type* and the *ExternalTypeName* is created with this command. The name of the respective type is used to check whether this already exists. If it already exists, the command is aborted.

#### Command parameters

| Parameter | |
|---|---|
| Type | The *object type* of the element that is to be created is specified here. This can be a "DEVICE", a "TO_by_Filter", or a "SINAMICS_CU320_STATION". |
| Name | Name of the new object |
| Version | Version of the new object. E.g.: "V4_5_0" |

| | |
|---|---|
| ExternalTypeName | Type of TO or device to be imported<br><br>Examples of devices:<br>"SIMOTION_D425", "SIMOTION_D445" ,<br>"SINAMICS_S120", etc.<br>You can find a list of all possible values by searching the SCOUT online help for *ExternalTypeName*.<br><br>Examples of TO:<br>"SIMOTION_CAM", "SIMOTION_POSAXIS_REAL", "SIMOTION_DRIVEAXIS_REAL", etc.<br>If specified as "TO_by_Filter" type, the name of the corresponding TO export must be specified for this parameter (without ".XML"). |
| ReferenceObject | Optional: Transfer of a reference object (list box) to directly update the display of the user interface. |
| CPUInterface | Optional: The bus system of the device. The following values are possible here: DP1, DP2/MPI, PN1, DP_INT |
| FilePath | The path to the export of the device. |
| UseLocalPath | Optional: Specifies whether the relevant export is located in the relevant subdirectory of the ProjectGenerator. If this is the case, only the name of the export need be specified. |
| IP_Address | Optional: IP address of the device |
| Address | Optional: The address of the device. This is a PROFIBUS address, so "NEXT_FREE" can be specified here. The ProjectGenerator automatically determines the next free address. |
| ObjectAddress | Optional: If a TO is being imported, the object address can be specified using this parameter. |
| Filter | Optional: This parameter must be used if the type has been specified as "TO_by_Filter". For this parameter, the name of the folder beneath the path which contains the exports of the TOs to be imported must be specified..<br>Example:<br>Path SIMOTION\TechnologyObjects\V4_3<br>Name of the folder SIMOTION_AXES |

The table below provides an overview of which parameters can be used with which type.

Table 2- 48 Overview of type/parameter combinatorics

| Type | DEVICE | TO_by_Filter | SINAMICS_CU320_STATION |
|---|---|---|---|
| Name | X | X | X |
| Version | X | | X |
| External-TypeName | X | X | X |
| ReferenceOb-ject | X | X | X |
| CPUInterface | | | X |
| FilePath | | X | X |
| UseLocalPath | | | X |
| IP_Address | | | X |
| Address | | | X |
| Ob-jectAddress | | X | |
| Filter | | X | |

## Example

Table 2- 49 Code example

```
<!--  Add a SIMOTION D445 to the project, refresh the listbox(LB_Devices) -->
<Command ID="100" Name="CreateObject" NCID="20">
    <Parameter Type="Device"
        Name="D445"
        Version="V4_3"
        ExternalTypeName="SIMOTION_D445"
        ReferenceObject="LB_Devices"/>

<!--  Add a new axis to the project, get the parameters (Name, Version etc...) from the
controls (textbox ,comboboxes)and refresh the listbox(LB_Axes)  -->

    <Parameter Type="TO"
        Name="__ref_TB_Name_New_Axis.Text"
        ExternalTypeName="__ref_CB_Name_New_Axis_ExternalTypeName.Text"
        ReferenceObject="LB_Axes"
        Filter="SIMOTION_AXES"/>
</Command>
```

## 2.3.2 DeleteObject

A newly created object can be deleted with this command. Objects that already exist in a project cannot be deleted.

### Command parameters

| Parameter | |
|---|---|
| Type | The *object type* of the element that is to be deleted is specified here. This can be a *device,* a *TO*, a *SinamicsDevice* or a *Sinamics DO*. |
| Name | Name of the object to be deleted. |
| ExternalTypeName | Type of the object to be deleted. |
| ReferenceObject | Transfer of a reference object (list box) to directly update the display of the user interface. |

### Example

Table 2- 50 Code example

```
<Command
    ID="100"
    Name="DeleteObject"
    NCID="20">
    <Parameter
        Type="Device"
        Name="__ref_TB_Name_New_Device.text"
        ExternalTypeName="__ref_CB_Name_New_Device_ExternalTypeName.text"
        ReferenceObject="LB_Devices"/>
    <Parameter
        Type="TO"
        Name="__ref_TB_Name_New_Axis.text"
        ExternalTypeName="__ref_CB_Name_New_TO_ExternalTypeName.text"
        ReferenceObject="LB_Axes"/>
</Command>
```

## 2.3.3 ImportIOTable

Address tables can be imported into the selected device with this command.

Any number of address tables can be imported with one call of the command.
The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Path | Path to the export file. The path can also be specified relatively. |
| Type | The parameters can have the following values:<br>"XML": Import of an I/O table in SIMOTION export format |

### Example

Table 2- 51 Code example

```
<Command
    ID="10"
    Name="ImportIOTable"
    NCID="20">
    <Parameter
        Path="SIMOTION\EquipmentModules\V4_3\Module1\IOTable.xml"
        Type="XML"/>
</Command>
```

## 2.3.4 ImportLibrary

Libraries can be imported into the project with this command.

Any number of libraries can be imported with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the library in the project.<br>The value can also be assigned a reference to a user interface element. The *Text* property of the element is evaluated here. |
| Path | Path to the XML export of the library. The path can also be specified relatively. |
| ForceImport | Optional: If this parameter is set to True, a version check is not performed during the import if this library already exists in the project, and the library is replaced. |

### Example

Table 2- 52 Code example

```
<Command
    ID="1"
    Name="ImportLibrary"
    NCID="20">
    <Parameter
        Name="LDPV1"
        Path="SIMOTION\EquipmentModules\V4_3\LDPV1_Library\LDPV1.xml"/>
    <Parameter
        Name="LMsgHdl"

Path="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\XML\LMsgHdl\LMsgHdl.xml"/>
</Command>
```

## 2.3.5 ImportScript

A script is imported from a specified path to a project with this command. It can be integrated in arbitrary object levels by specifying the *Destination* parameter.

### Command parameters

| Parameter | |
|---|---|
| Name | New name of the script |
| Path | Import path of the script, including the script name |
| Device | Target device (omitted with project level) |
| Destination | Target object below the device |
| ForceImport | You can set whether the script import is mandatory here. |

### Example

Table 2- 53 Code example

```
<Command
    ID="100"
    Name="ImportScript"
    NCID="20">
    <Parameter
        Name="AxisSimulationOnOff"
        Path="SIMOTION\Scripts\V4_3\AxisSimulationOnOff.txt"
        Device="D435"
        Destination="Axis_1"
        ForceImport ="False"/>
</Command>
```

## 2.3.6 ImportScriptsToProject

All exported scripts in the specified path are imported on the project level with this command.

### Command parameters

| Parameter | |
|---|---|
| Path | Path containing the scripts. |
| ForceImport | You can set whether the import of the script is mandatory here. |

### Example

Table 2- 54 Code example

```
<Command
    ID="100"
    Name="ImportScriptsToProject"
    NCID="20">
    <Parameter
        Path="SIMOTION\Scripts\V4_3"/>
</Command>
```

## 2.3.7 ImportTO

A new technology object is imported with this command.

### Command parameters

| Parameter | |
|---|---|
| Name | New name of the TO. |
| Path | Path to the export file. If no path is specified, the *ExternalTypeName* applies. |
| ExternalTypeName | Type name (name of the export file). If no *ExternalTypeName* is specified, the XML file of the TO export must be specified. |
| ParentTO | Optional: To import a subordinate TO, specify the name of the TOs to which the new TO should be imported here. |
| ObjectAddress | Optional: The object address of the TO can be set by specifying this parameter. If this parameter is not used, SCOUT assigns a free object address. |

### Example

Table 2- 55 Code example

```
<Command
    ID="100"
    Name="ImportTO"
    NCID="20">
    <Parameter
        Name="__ref_TB_Name_New_Axis.text"
        ExternalTypeName="SIMOTION_GEARAXIS"/>
</Command>
```

## 2.3.8 ImportUnit

Units can be imported into the active device with this command.

Any number of units can be imported with one call. The parameter line can be filled in several times for this purpose.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the unit in the project. The value can also be assigned a reference to a user interface element. The *Text* property of the element is evaluated here. |
| Path | Path to the unit. The import type (ST or XML) is set automatically depending on the file extension. |
| ForceImport | You can set whether the unit import is mandatory here. An already existing unit is always overwritten. |
| ObjectAddress | Optional: The object address of the program can be set by specifying this parameter. If this parameter is not used, SCOUT assigns a free object address. |

### Example

Table 2- 56 Code example

```
<Command
    ID="1"
    Name="ImportUnit"
    NCID="20">
    <Parameter
        Name="pLMsgHdl"
        Path="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\pLMsgHdl.XML"
        ForceImport="True"/>
    <Parameter
        Name="__ref_TB_new_unit.text"
        Path="SIMOTION\EquipmentModules\V4_3\IntelligentBelt_LIBelt\Units\pLIBelt.st"
        ForceImport="False"/>
</Command>
```

## 2.3.9 ImportUnitInLib

A unit is imported into a library with this command. If several units are also to be imported into different libraries, the parameter line can be called several times. Units can also be imported into existing libraries.

### Command parameters

| Parameter | |
|---|---|
| Name | New name of the unit |
| LibName | Name of the target library |
| ForceImport | Mandatory import of the unit, i.e. overwriting of a unit with the same name |
| Path | Path to the export of the unit |
| Mode | XML import |

### Example

Table 2- 57 Code example

```
<Command
    ID="100"
    Name="ImportUnitInLib"
    NCID="20">
    <Parameter
        Name="fMsgHdl"
        LibName="LMsgHdl"/>
        ForceImport="False"
        Path="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\pLMsgHdl.XML"
        Mode="XML"/>
</Command>
```

## 2.3.10 ImportWatchTable

A watch table is imported into the project with this command.
If variables exist within the watch table, the path to a TO (TO name) and a unit (unit name) can each be changed to a new path. Several units or TOs cannot be readdressed.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the watch table |
| Path | Path to the export file |
| ForceImport | Replace if the watch table already exists |
| Unit | Name of the new unit (optional) |
| TO | Name of the new TO (optional) |

### Example

Table 2- 58 Code example

```
<Command
    ID="100"
    Name="ImportWatchTable"
    NCID="20">
    <Parameter
        Name="Control_Watch"
        Path="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHdl\Watch.XML"
        ForceImport="True"
        Unit="Unit2"
        TO ="Axis_1"/>
</Command>
```

## 2.3.11 AddGlobalVariable

Global device variables can be inserted into the selected device with this command.

Any number of global device variables can be inserted with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the variable. |
| DataType | Data type of the variables. |
| IsRetain | Optional: Retain property of the variables. |
| FieldLength | Optional: Specification of the length of the variable (Fieldlength >1 means Array[0 to Fieldlength-1] of Datatype) |

**Subparameter**

| Comment | |
|---|---|
| LanguageID | DE, EN, FR, IT, ES |
| Comment | Comment of the variable in the selected language. |

**Subparameter**

| InitValue | |
|---|---|
| Value | Initial value of the variables. |
| Index | Optional: Index of the initial value in case the variable is an array. |
| Convert | HEX |

## Example

Table 2- 59 Code example

```
<Command
      ID="12"
      Name="AddGlobalVariable">
      <Parameter
            Name="Value1"
            DataType="WORD"
            FieldLength=""
            IsRetain="False" >
            <InitValue
                  Value="0"
                  Convert="HEX"/>
            <Comment
                  LanguageID="DE"
                  Comment="Kommentar" />
            <Comment
                  LanguageID="EN"
                  Comment="Comment" />
      </Parameter>
      <Parameter
            Name="Value2"
            DataType="WORD"
            FieldLength="2"
            IsRetain="False" >
            <InitValue
                  Value="1"
                  Index=""
                  Convert="HEX"/>
            <InitValue
                  Value="2"
                  Index=""
                  Convert="HEX"/>
            <Comment
                  LanguageID="DE"
                  Comment="Kommentar" />
            <Comment
                  LanguageID="EN"
                  Comment="Comment" />
      </Parameter>
</Command>
```

## 2.3.12 ImportGlobalVariableTable

With this command, XML exports of global variables can be imported into the selected device.

Any number of XML exports can be imported with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Path | Path to the XML export. |
| Type | XML/CSV (CSV not yet implemented). |

### Example

Table 2- 60 Code example

```
<Command
      ID="12"
      Name="ImportGlobalVariableTable">
      <Parameter
            Path="Path\To\XMLExport.XML"
</Command>
```

## 2.3.13 ImportAlarmSTable

With this command, XML exports of AlarmS messages can be imported into the selected device.

Any number of AlarmS messages can be imported with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Path | Path to the XML export. |
| Replace | Optional: If they have the same name, the present message can be replaced with the new one. The default is FALSE. |

### Example

Table 2- 61 Code example

```
<Command
      ID="12"
      Name="ImportAlarmSTable">
      <Parameter
            Path="Path\To\XMLExport.XML"
            Replace="False"/>
</Command>
```

## 2.3.14 AddAlarmSMessage

AlarmS messages can be inserted into the selected device with this command.

Any number of AlarmS messages can be inserted with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Symbolic name of the message. |
| MessageType | Type of message:<br>FAULT: Fault message<br>INFO: Operating message |
| Class | Message class (0 to 15). The display class is required if a definite choice of specific messages is requested for an operator panel during the configuration of the display. |
| AlarmID | Number of the message (optional). |
| DoPrint | Set the parameter to True if this message (Alarm_S) is to be simultaneously printed at the connected printer when it is displayed on the HMI device (optional). |

**Subparameter (optional)**

**Text**

| | |
|---|---|
| LanguageID | Setting of the language for the following text: DE, EN, FR, IT, ES.<br>The abbreviation can be used for these languages.<br>Other languages can be selected by specifying the corresponding Microsoft LocaleID number. Information about LocaleID can be found at: http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx (http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx) |
| MessageText | Message text |
| InfoText | Infotext |

## Example

Table 2- 62 Code example

```
<Command ID="12" Name="AddAlarmSMessage">
      <Parameter Name="Fault_1"
            MessageType="FAULT"
            Class="0"
            AlarmID="0"
            DoPrint="False">
            <Text LanguageID="DE"
                  MessageText="This is Fault 1."
                  InfoText="This is a Infotext!"/>
      </Parameter>
</Command>
```

## 2.3.15 AddIOVariable

I/O variables can be inserted into the selected device with this command.

Any number of I/O variables can be inserted with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| | |
|---|---|
| **Parameter** | |
| Name | Name of the variable. |
| Address | Target address of the I/O variables (e.g., PIW256). |
| DataType | Data type of the variable |
| FieldLength | Optional: Specification of the variable length.<br>(Fieldlength >1 means Array[0 to Fieldlength-1] of Datatype). |
| IsReadOnly | Optional: Read-only property of the variables. |
| ProcessImageTask | Optional: Specification of the update cycle. |
| SubstitutionStrategy | Specification of the substitute value strategy.<br>"CPU_STOP" - CPU goes to STOP<br>"LAST_VALUE" - last value<br>"SUBS_VALUE" - the substitute value in<br>subparameter substitution applies |
| **Subparameter (optional)** | |
| **Comment** | |
| LanguageID | DE, EN, FR, IT, ES |
| Comment | Comment of the variable in the selected language. |
| **Subparameter** | |
| **Substitution** | |
| Value | Substitute value of the variable |
| Index | Optional: Index of the substitute value in case the variable is an array. |

## Example

Table 2- 63 Code example

```
<Command
      ID="12"
      Name="AddIOVariable">
      <Parameter
            Name="Value1"
            Address="PQW500"
            DataType="WORD"
            FieldLength=""
            IsReadOnly="False"
            ProcessImageTask=""
            SubstitutionStrategy="SUBS_VALUE">
                  <Comment
                        LanguageID="DE"
                        Comment="Kommentar"/>
                  <Comment
                        LanguageID="EN"
                        Comment="Comment" />
                  <Substitution
                        Value="0"/>
      </Parameter>
      <Parameter
            Name="Value2"
            Address="PQW600"
            DataType="WORD"
            FieldLength="2"
            IsReadOnly="False"
            ProcessImageTask=""
            SubstitutionStrategy="SUBS_VALUE">
            <Comment
                  LanguageID="DE"
            Comment="Comment" />
            <Comment
                  LanguageID="EN"
                  Comment="Comment" />
            <Substitution
                  Value="1"
                  Index="0"/>
            <Substitution
                  Value="1"
            Index="1"/>
      </Parameter>
</Command>
```

## 2.3.16 RemoveTO

An existing technology object (TO) can be deleted with this command. If several technology objects are to be deleted, the parameter line can be specified several times.

### Command parameters

| Parameter | |
|---|---|
| Device | Name of the SIMOTION device on which the TO is located. |
| TO | Name of the TO. |

### Example

Table 2- 64 Code example

```
<Command ID="100" Name="RemoveTO">
     <Parameter
          Device="NewDevice"
          TO="Axis_1"
     />
</Command>
```

## 2.3.17 RemoveUnit

An existing unit can be deleted with this command. Programs assigned to the execution system must be removed by the user with the RemoveProgram command.

If several units are to be deleted, the parameter line can be specified several times.

### Command parameters

| Parameter | |
|---|---|
| Device | Name of the SIMOTION device in which the unit is located. |
| Name | Name of the unit. |

### Example

Table 2- 65 Code example

```
<Command ID="101" Name="RemoveUnit">
     <Parameter
          Device="NewDevice"
          Name="xPTemplAxisFB"
     />
</Command>
```

## 2.3.18 RemoveLibrary

An existing library can be deleted with this command. If several libraries are to be deleted, the parameter line can be specified several times.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the library. |

### Example

Table 2- 66 Code example

```
<Command ID="102" Name="RemoveLibrary">
     <Parameter Name="LMCBasic" />
</Command>
```

## 2.3.19 RemoveIOVariable

An I/O variable can be deleted from a selected device with this command. If several I/O variables are to be deleted, the parameter line can be specified several times.

### Command parameters

| Parameter | |
|---|---|
| Device | Name of the SIMOTION device on which the I/O variable is located. |
| Name | Name of the I/O variable. |

### Example

Table 2- 67 Code example

```
<Command ID="103" Name="RemoveIOVariable">
     <Parameter
          Device="NewDevice"
          Name="testIOVariable"
     />
</Command>
```

## 2.3.20 RemoveWatchtable

An existing watch table can be deleted from the project with this command. If several watch tables are to be deleted, the parameter line can be specified several times.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the watch table. |

### Example

Table 2- 68 Code example

```
<Command ID="104" Name="RemoveWatchtable">
     <Parameter
          Name="watchtable_1"
     />
</Command>
```

## 2.3.21 RemoveGlobalDeviceVariable

Global variables can be deleted with this command. If several variables are to be deleted, the parameter line can be specified several times. The command is executed for the device being processed at the time of the execution.

### Command parameters

**Parameter**

| | |
|---|---|
| Name | Name of the global variable to be deleted. |

### Example

Table 2- 69 Code example

```
<Command ID="13" Name="RemoveGlobalDeviceVariable" NCID="14">
     <Parameter Name="var1" />
</Command>
```

# 2.4 Information commands

## 2.4.1 GetAvailableAxesByType

All axes of the transferred type of the active device that are either in the project or in the associated ProjectGenerator database are returned with this command. If no type is transferred, all axes are returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

When called in the source code, a collection (Arraylist) of *SortedDictionary* objects is returned. These offer two keys for accessing the *ExternalTypeName* (EXTERNALTYPENAME) and the name (NAME) of the axis.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Type | The *ExternalTypeName* of SIMOTION SCOUT is transferred here. |

### Example

Table 2- 70 Code example

**Transfer to a control**

```
Source="__call_GetAvailableAxesByType('SIMOTION_GEARAXIS')"
```

**Use in the source code**

```
REM Get all axes in the active device
Dim AxesList As System.Collections.Arraylist = My-
App.myIsl._GetAvailableAxesByType()

For Each tmpAxis As System.Collections.Generic.SortedDictionary(Of String,
String) in AxesList

          If tmpAxis('EXTERNALTYPENAME') = 'SIMOTION_GEARAXIS' THEN
                    REM Do something if it’s a gear axis
          End IF

          If tmpAxis('NAME') = 'MasterAxis' THEN
                    REM Do something if the axis name is MasterAxis
          End IF


Next
```

## 2.4.2 GetAvailableAxesByVersion

All axes that are available for import are returned with this command. This command can be transferred directly to a control property via the *__Call_* prefix. When called in the source code, a collection (*Arraylist*) of *SortedDictionary* objects is returned. These offer a "NAME" key for accessing the name of the axis.

### Command parameters

#### Transfer parameters

| | |
|---|---|
| Version | Specification of the main version of the device (e.g. V4_4). |

### Example

Table 2- 71 Code example

```
Source="__call_GetAvailableAxesByVersion(__call_getVersionOfActualDevice)"
```

## 2.4.3 GetAvailableDevices

All devices that are available in a directory for import are returned with this command. This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

#### Transfer parameters

| | |
|---|---|
| Version | Specification of the main version of the device (e.g. V4_2, V4_3, V4_4). |

### Example

Table 2- 72 Code example

**Transfer to a control**

```
Source="__call_GetAvailableDevices('V4_3')"
```

**Use in the source code**

```
REM Get all devices with version 4.2 that can be imported
Dim DeviceList As System.Collections.Arraylist = My-
App.myIsl.GetAvailableDevices('V4_2')

REM Add the names of the devices to a listbox named LB_Devices
For Each tmpDevice As System.Collections.Generic.SortedDictionary(Of
String, String) in DeviceList
          @LB_Devices@.Items.Add(tmpDevice('NAME').ToString())
Next
```

## 2.4.4 GetAvailableDeviceVersions

All versions of the device that are available for import are returned with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of SortedDictionary objects with the unique "NAME" key.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 73 Code example

**Transfer to a control**

```
Source="__call_GetAvailableDeviceVersions"
```

**Use in the source code**

```
REM Get all available device versions
Dim VersionList As System.Collections.Arraylist = My-
App.myIsl.GetAvailableDeviceVersions()

REM Add the available device versions to a combobox named CMB_Versions
For Each tmpVersion As System.Collections.Generic.SortedDictionary(Of
String, String) in VersionList
         @ CMB_Versions @.Items.Add(tmpVersion ('NAME').ToString())
Next
```

## 2.4.5 GetDevicesInProject

All SIMOTION devices (no SINAMICS) that are either in the project or in the ProjectGenerator are returned with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the following keys:

NAME, OBJECTID, TYPEID, TYPENAME, and VERSION

### Command parameters

This command does not have any parameters.

### Example

Table 2- 74 Code example

**Transfer to a control**

```
Source="__call_GetDevicesInProject"
```

**Use in the source code**

```
REM Get all devices in the project
Dim DeviceList As System.Collections.Arraylist = My-
App.myIsl.GetDevicesInProject()

REM Objects to represent a ListViewItem and its SubItems
Dim DeviceItem As ListViewItem
Dim ParameterItem As ListViewItem.ListViewSubItem

REM Start updating the ListView
@LV_Devices@.BeginUpdate()

REM Add all information on a device to the listView named LV_Devices
For Each tmpDevice As System.Collections.Generic.SortedDictionary(Of
String, String) in DeviceList
         DeviceItem = @LV_Devices@.Items.Add(tmpDevice('NAME').ToString())
         ParameterItem = DeviceItem.SubItems.
Add(tmpDevice('VERSION').ToString()
         ParameterItem = DeviceItem.SubItems.
Add(tmpDevice('TYPENAME').ToString())
         ParameterItem = DeviceItem.SubItems.
Add(tmpDevice('OBJECTID').ToString())
         ParameterItem = DeviceItem.SubItems.
Add(tmpDevice('TYPEID').ToString())

Next

REM End updating the ListView
@LV_Devices@.EndUpdate()
```

The ListView control is defined as follows:

```
<Control Action="add"
 Type="ListView"
 Name=" LV_Devices"
 View="Details"
 MultiSelect="False"
 FullRowSelect="True"
 HideSelection="True"
 Location="190, 240"
 Size="700, 370"
 GridLines ="True">

 <ListViewItem Name="Version"
 size="100"
 alignment="HorizontalAlignment.Left" >
 <ListViewSubItem Name="Version"
 size="100"
 alignment="HorizontalAlignment.Left" >
 <ListViewSubItem Name="Typename"
 size="100"
 alignment="HorizontalAlignment.Left" >
 <ListViewSubItem Name="ObjectID"
 size="200"
 alignment="HorizontalAlignment.Left" >
 <ListViewSubItem Name="TypeID"
 size="200"
 alignment="HorizontalAlignment.Left" >

 </ListViewItem>
 <Events>
 <SelectedIndexChanged code="" >
 </Events>
</Control>
```

## 2.4.6 GetSimotionDeviceName

The name of the active device is returned with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it from the source code returns the name as a string.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 75 Code example

**Transfer to a control**

```
Text="__call_GetSimotionDeviceName"
```

**Use in the source code**

```
Dim strDeviceName As String = MyApp.myIsl.GetSimotionDeviceName()
```

## 2.4.7 GetUnits

All existing units in the project, e.g. for display in a list box, are returned with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. If the function is called in the source code, it delivers a collection of SortedDictionary objects with the following keys:

NAME, OBJECTID, TYPEID, and TYPENAME.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 76 Code example

**Transfer to a control**

```
Source="__call_GetUnits"
```

**Use in the source code**

```
REM Get all units in the project
Dim UnitsList As System.Collections.Arraylist = MyApp.myIsl.GetUnits()

REM loop through all units and find the unit MyUnitName
For Each tmpUnit As System.Collections.Generic.SortedDictionary(Of String,
String) in UnitsList
If tmpUnit('NAME').ToLower = 'myunitname' Then

         REM write the name of the unit in a TextBox
         @TB_UnitName@.Text = tmpUnit('NAME')

         REM call a command that does something with the unit e.g.
         REM GetValueOfUnitVariable
         MyApp.NextCommand(400)

         REM exit the for loop after finding the unit
         Exit For
End IF
Next
```

## 2.4.8 GetValueOfLibVariable

The currently valid value of a variable or constant is read from a library unit and returned as a string with this command. If a value is not found either in the project or in the database, the *DefaultValue* is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Name | Name of the variable or constant |
| Library | Name of the library |
| Unit | Name of the unit |
| DefaultValue | Substitute value when no value is found |

### Example

Table 2- 77 Code example

**Transfer to a control**

```
Text="__call_GetValueOfLibVariable(LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES,LMsgH
dl,cPublic,100)"
```

**Use in the source code**

```
Dim LngthOfMsgs As String =
My-
App.myISL._GetValueOfLibVariableMyApp.myISL._GetValueOfLibVariable('LMSGHD
L_LENGTH_OF_ACTIVE_MESSAGES','LMsgHdl','cPublic',100)
```

## 2.4.9 GetValueOfUnitVariable

This command reads out the currently valid value of a variable or constant from a unit and returns it as a string. If a value is not found either in the project or in the database, the *DefaultValue* is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Name | Name of the variable or constant |
| Unit | Name of the unit |
| DefaultValue | Substitute value when no value is found |

### Example

Table 2- 78 Code example

**Transfer to a control**

```
Text="__call_GetValueOfUnitVariable(PTEMPL_NUMBER_OF_EMS_IN_1ST_LAYER,dGlo
balPTempl,3)"
```

**Use in the source code**

```
Dim NumberOfEMS As String = My-
App.myISL._GetValueOfUnitVariable('PTEMPL_NUMBER_OF_EMS_IN_1ST_LAYER','dGl
obalPTempl',3)
```

## 2.4.10 GetConfigdataValueOfTO

This command reads out the currently valid value of a configuration date of an axis and returns it as a string. If a value is not found either in the project or in the database, the *DefaultValue* is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the TOs. |
| ConfigDataName | Name of the configuration date. |
| DefaultValue | Default value, if this configuration date was not found in the database or the project. |

### Example

Table 2- 79 Code example

**Transfer to a control**

```
Text="__call_GetConfigdataValueOfTO(TO_Axis_Red,TypeOfAxis.NumberOfDataSets.DataSet_1.Gear
.numFactor,1)"
```

**Use in the source code**

```
Dim numFactor As String = MyApp.myISL._GetConfigdataValueOfTO
('TO_Axis_Red','TypeOfAxis.NumberOfDataSets.DataSet_1.Gear.numFactor',1)
```

## 2.4.11 GetValueOfTemporaryVariable

The value of a temporary variable is returned as a string with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. The returned value can also be converted to other data types.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the temporary variables. |
| Type | You can choose between the options LOCAL and GLOBAL. |
| Lokale | Temporary variables are deleted automatically when the next module is called; global variables are retained. If this parameter is not supplied, LOCAL is assumed automatically. |
| DefaultValue | Default value of the temporary variables if they are not found in the data management. |

**Note**

Errors in conversion may occur when working with floating-comma numbers. See the code example below.

## Example

Table 2- 80 Code example

**Transfer to a control**

```
Text="__call_ GetValueOfTemporaryVariable(Hello,LOCAL,World_Not_Found)"
```

**Use in the source code**

```
REM define different global variables
MyApp.myISL.SetTemporaryVariable('DoubleValue','Global',3601.8)
MyApp.myISL.SetTemporaryVariable('DoubleValueAsStringWithDot','Global', '3601.8')
MyApp.myISL.SetTemporaryVariable('DoubleValueAsStringWithComma','Global', '3601,8')
MyApp.myISL.SetTemporaryVariable('StringValue','Global', 'SomeText')

Dim DoubleVariable As Double
Dim StringVariable As String

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValue' , 'Global', -1)
MsgBox (DoubleVariable)   REM No error and output correct (3601.8)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithComma',
'Global', -1)
MsgBox (DoubleVariable)   REM No error and output  correct (3601.8)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithDot',
'Global', -1)
MsgBox (DoubleVariable)   REM No error but output not correct (36018)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('StringValue', 'Global', -1)
MsgBox (DoubleVariable)   REM  error converting the String to a Double value

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValue', 'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error and output correct(3601.8)

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithDot',
'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error but output not correct(36018)

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithComma',
'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error and output correct(3601.8)
```

## See also

SetTemporaryVariable (Page 50)

## 2.4.12 GetAvailableObjectsByVersionByFilter

All available exports in the subdirectories of the TechnologyObjects folder which are available for import are returned with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

| Parameter | |
|---|---|
| DeviceVersion | Specification of the main version of the device. |
| Filter | Specification of the subdirectory to be searched. |

### Example

Table 2- 81 Code example

**Transfer to a control**

```
Source="__call_getAvailableObjectsByVersionByFilter(__call_getVersionOfActualDevice,'SIMOT
ION_AXES')"
```

**Use in the source code**

```
REM Get all CAM Exports in the directory \TechnologyObjects\V4_3\SIMOTION_CAMS
Dim CAMExports As System.Collections.Arraylist =
MyApp.myIsl.GetAvailableObjectsByVersionByFilter('V4_3','SIMOTION_CAMS')
```

## 2.4.13 GetVersionOfActualDevice

The current version of the active device is returned as a string with this command.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 82 Code example

**Transfer to a control**

```
Text="__call_ GetVersionOfActualDevice()"
```

**Use in the source code**

```
Dim strActualVersion As String = MyApp.myIsl._GetVersionOfActualDevice()
MsgBox(strActualVersion)  REM  Output is 'V4_3'
```

## 2.4.14 GetAvailableTOsByType

All TOs of the transferred type of the active device that are either in the project or in the associated ProjectGenerator database are returned with this command. If no type is transferred or the "All" string is transferred, all TOs are returned.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the "EXTERNALTYPENAME" and "NAME" keys.

### Command parameters

| Parameter | |
|---|---|
| Filter | The *ExternalTypeName* of the SIMOTION SCOUT is transferred here, or specified ALL in order to report them all back. |

### Example

Table 2- 83 Code example

**Transfer to a control**

```
Source="__call_GetAvailableTOsByType('SIMOTION_GEARAXIS')"
```

**Use in the source code**

```
REM Get all TOs in the project
Dim TOsList As System.Collections.Arraylist = MyApp.myIsl._GetAvailableTOsByType()
REM Or
REM Dim TOsList As System.Collections.Arraylist = My-
App.myIsl._GetAvailableTOsByType('All')
REM Get all CAMs in the project
Dim CAMsList As System.Collections.Arraylist = My-
App.myIsl._GetAvailableTOsByType('SIMOTION_CAM')
```

## 2.4.15 GetActualIPofDevice

The IP address of the interface of the active device connected to "Ethernet(1)" is returned as a string with this command.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

| Parameter | |
|---|---|
| IPIndex | -1 supplies the whole IP address |
| | 0-3 supplies the selected part of the IP address |

### Example

Table 2- 84 Code example

**Transfer to a control**

```
Text="__call_GetActualIPofDevice (-1)"
```

**Use in the source code**

```
REM write the complete IP Address in a Textbox
@TB_IPAdress@.Text = MyApp.myIsl.GetActualIPofDevice()

REM write the parts of the IP Address in 4 textboxes
@TB_IPAdress1@.Text = MyApp.myIsl.GetActualIPofDevice(0)
@TB_IPAdress2@.Text = MyApp.myIsl.GetActualIPofDevice(1)
@TB_IPAdress3@.Text = MyApp.myIsl.GetActualIPofDevice(2)
@TB_IPAdress4@.Text = MyApp.myIsl.GetActualIPofDevice(3)
```

## 2.4.16 GetSimotionDeviceType

The type of the active SIMOTION device is returned as a string with this command.

This command can be transferred directly to a control property via the *__Call_* prefix. The *ExternalTypeName* of the device is returned.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 85 Code example

**Transfer to a control**

```
Text="__call_GetSimotionDeviceType"
```

**Use in the source code**

```
REM Get the Type of the actual device
Dim strDeviceType As String = MyApp.myIsl.GetSimotionDeviceType()
MsgBox(strDeviceType) REM Output is 'SIMOTION_D445'
```

## 2.4.17 GetSystemVariableValueOfTO

This command reads out the currently valid value of a system variable of an axis and returns it as a string. If a value is not found either in the project or in the database, the *DefaultValue* is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the TOs. |
| SystemVariableName | Name of the system variables. |
| DefaultValue | Default value, if this system variable was not found in the database or the project. |

### Example

Table 2- 86 Code example

**Transfer to a control**

```
Text=__call_GetSystemVariableValueOfTO(TO_Axis_Red,defaultAdditiveTorque,100)"
```

**Use in the source code**

```
REM Get the Type of the actual device
@TB_AdditiveTorque@.Text = MyApp.MyIsl._GetSystemVariableValueOfTO('TO_Axis_Red', 'de-
faultAdditiveTorque',100)
```

## 2.4.18 GetInputAddressOfX142

The input address of the X142 interface on a SIMOTION D4x5-2 can be read with this command. This function can only be used with SIMOTION SCOUT as of Version 4.4.

This command can be transferred directly to a control property via the __Call_ prefix.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 87 Code example

```
Text="__call_GetInputAddressOfX142"
```

## 2.4.19 GetOutputAddressOfX142

The output address of the X142 interface on a SIMOTION D4x5-2 can be read with this command. This function can only be used with SIMOTION SCOUT as of Version 4.4.

This command can be transferred directly to a control property via the __Call_ prefix.

### Command parameters

This command does not have any parameters.

### Example

Table 2- 88 Code example

```
Text="__call_GetOutputAddressOfX142"
```

## 2.4.20 GetDeviceProperty

The value of every permissible HW Config property (attribute) of a device can be read out with this command. If it is the property of a module or a submodule, then the appropriate module (and submodule) number must be specified. As it may be that the required property is within a field, as with *LocalInAddresses* or *LocalOutAdresses*, it is also possible to optionally specify the name of the relevant object with *ObjectType*.

The command can only be executed within the VB code.

### Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the device. |
| Name | Name of the HW Config property/attribute. |
| SlotNumber | Optional: Specification of the slot number for the addressing of subordinate slots. |
| SubSlotNumber | Optional: Specification of the subslot number for the addressing of subordinate slots. |
| ObjectType | Optional: Specification of the object that is to be accessed, e.g. *LocalInAddresses* or *LocalOutAddresses*. |

### Example

Table 2- 89 Code example

```
<Control
     Action="add"
     Type="Button"
     Name="BT_GetDeviceProperty"
     Text="GetDeviceProperty"
     Location="280, 360"
     Size="150, 30"
     Enabled="true"
     Visible="true"
     ToolTip="Test the function GetDeviceProperty">
     <Events>
         <Click code="
          DIM String_Value =
          MyApp.myIsl.GetDeviceProperty('MyET200', 'LogicalAddress', '2',
'6', 'LocalInAddresses')
          "
         />
     </Events>
</Control>
```

# 2.5 Commands for the source and object manipulation

## 2.5.1 Commands for the source manipulation

### 2.5.1.1 SetDeviceAndSlaveInformation

Different information from the device and the I/O is integrated in a unit between two labels (start and end labels, compare generic areas for *SetValue*) with this command. The unit must be inserted with *ImportUnit*.

All types of information can be imported into this area via multiple calls of the parameter structure. If the information is to be imported into different areas, the entire command must be called several times.

#### Command parameters

**Command tags (cannot be called generically)**

| | |
|---|---|
| StartLabel | Start of the area to be replaced. |
| EndLabel | End of the area to be replaced. |
| Unit | Unit in which the area is replaced before the import. |

**Parameter**

| | |
|---|---|
| Path | Specifies the path for the code fragment for the generic area import. |
| Type | The parameters can have the following values:<br>CyclicDOs<br>NonCyclicDOs<br>Peripherals<br>DOsWithCyclicStandTelIO<br>AdditionObjects<br>Axes<br>Cams<br>CamTracks<br>Controllers<br>ExternalEncoders<br>FixedGears<br>FOOs<br>Formulas<br>Measureswitches<br>OutputCams<br>PathObjects<br>Sensors<br>TemperatureControllers |

## Labels for the Type parameters

The following labels are automatically replaced in the individual types:

| Type | Label | Replaced with: |
|---|---|---|
| **CyclicDOs** | | |
| | <DOName> | The name of the *DriveObject* is entered here. |
| | <counter> | The current index counter is entered here. |
| | <LogAddress> | The logical input or output address is entered here. |
| | <IOID> | Whether the logical address of the input or the output is entered here. |
| | <TelegrammType39x> | Whether the message frame used is a 39x message frame is entered here. |
| **NonCyclicDOs** | | |
| | <DOName> | The name of the *DriveObject* is entered here. |
| | <counter> | The current index counter is entered here. |
| | <LogAddress> | The diagnostics address is entered here. |
| | <DONumber> | The number of the *DriveObject* is entered here. |
| **Peripherals** | | |
| | <counter> | The current index counter is entered here. |
| | <SystemID> | The subsystem ID of the slave is entered here. |
| | <Slaveaddress> | The slave address is entered here. |
| | <Devicename> | The associated device name is entered here. |
| **DOsWithCyclicStandTelIO** | | |
| | <counter> | The current index counter is entered here. |
| | <IOVariable> | The I/O variable that is entered on the standard message frame in the I/O list is entered here. |
| **AdditionObjects** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **Axes** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| | <TRUE_FALSE> | Identification of whether the axis has a *DriveObject*. |
| **Cams** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **CamTracks** | | |
| | <counter> | The current index counter is entered here. |

| Type | Label | Replaced with: |
|---|---|---|
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **Controllers** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **ExternalEncoders** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **FixedGears** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **FOOs** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **Formulas** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **Measureswitches** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **OutputCams** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **PathObjects** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **Sensors** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |
| **TemperatureContollers** | | |
| | <counter> | The current index counter is entered here. |
| | <TO_Name> | The current name of the technology object of the current type is entered here. |

## Example

Table 2- 90 Code example

```
<Command
    ID="3"
    Name="SetDeviceAndSlaveInformation"
    NCID="4"
    Unit="pStartupCheck"
    StartLabel="** start label project info **"
    EndLabel="** end label project info **">
    <Parameter
        Type="Peripherals"
        Path="SIMOTION\EquipmentModules\V4_3\StartupCheck\Data\
              Templates\PeripheralDeviceInfo.st"/>
    <Parameter
        Type="Axes"
        Path="SIMOTION\EquipmentModules\V4_3\StartupCheck\Data\
              Templates\TONameAxis.st"/>
    <Parameter
        Type="ExternalEncoders"
        Path="SIMOTION\EquipmentModules\V4_3\StartupCheck\Data\
              Templates\TONameExternalEncoder.st"/>
</Command>
```

The files under the path contain data which is to be written to the unit. The TONameAxis.st file for the axes type contains the following lines of code:

```
gasStartupCheckAxes[<counter>].toReference        := _to.<TO_Name>;
gasStartupCheckAxes[<counter>].sgTOName           := '<TO_Name>';
gasStartupCheckAxes[<counter>].boUsedInProject    := <TRUE_FALSE>;
```

Once the command has been called, these three lines are added to the pStartupCheck unit for each axis in the project, between the start and end labels. If the project contains, for example, a virtual axis "VirtAxis" and two real axes "Axis1" and "Axis2", then the area in the imported unit appears as follows:

```
// <<** start label project info **>>
gasStartupCheckAxes[0].toReference        := _to.Axis1;
gasStartupCheckAxes[0].sgTOName           := 'Axis1';
gasStartupCheckAxes[0].boUsedInProject    := TRUE;
gasStartupCheckAxes[1].toReference        := _to.Axis2;    gas-
StartupCheckAxes[1].sgTOName           := 'Axis2';
gasStartupCheckAxes[1].boUsedInProject    := TRUE;
gasStartupCheckAxes[2].toReference        := _to.VirtAxis;
gasStartupCheckAxes[2].sgTOName           := 'VirtAxis';
gasStartupCheckAxes[2].boUsedInProject    := False;
...
// <<** end label project info **>>
```

### 2.5.1.2 SetLabel

A label is replaced in a complete source (unit or library unit) with this command. The label is supplemented by a leading "<" and following ">" character.

As precondition for this command, an *ImportUnit* or *ImportLibrary* must be executed.

## Command parameters

| Parameter | |
|---|---|
| TargetName | Contains the name of a unit when *Type* is set to *Unit_Label*.<br>Contains the name of a library when *Type* is set to *Library_Label*. In this case, a valid unit name from this library must also be contained in the unit. |
| Unit | In *Unit*, only a unit name is specified if *Type* is set to *Library_Label*. |
| LabelName | Specifies the label that is replaced in the entire source. |
| Value | Specifies the value that is replaced. |
| Type | Library_Label, Unit_Label |

## Example

Table 2- 91 Code example

```
<Command   ID="100"
    Name="SetLabel"
    NCID="20">
    <Parameter
        TargetName="MyUnitName"
        Unit=""
        LabelName="UnitName"
        Value="MySelectedUnitName"
        Type="Unit_Label"/>
    <Parameter
        TargetName="MyUnitName"
        Unit=""
        LabelName="AxisName"
        Value="MySelectedAxisName"
        Type="Unit_Label"/>
    <Parameter
        TargetName="MyUnitName"
        Unit=""
        LabelName="CAMName"
        Value="MySelectedCAMName"
        Type="Unit_Label"/>
    <Parameter
        TargetName="MyUnitName"
        Unit=""
        LabelName="MasterName"
        Value="MySelectedMasterName"
```

```
        Type="Unit_Label"/>
</Command>
```

The imported unit "MyUnitName" contains the following code sequence:

```
gs<UnitName>.sPara.sTOs.toAxis :=  <AxisName>;
gs<UnitName>.sPara.sTOs.toCam  := <CAMName>;
gs<UnitName>.sPara.sTOs.toLeadingValue := <MasterName>;
```

Once the command has been called, the area appears as follows:

```
gsMySelectedUnitName.sPara.sTOs.toAxis :=  MySelectedAxisName;
gsMySelectedUnitName.sPara.sTOs.toCam  := MySelectedCAMName;
gsMySelectedUnitName.sPara.sTOs.toLeadingValue := MySelectedMasterName;
```

### 2.5.1.3 SetTaskInformationInUnit

Information, i.e. the existing (not deselected) execution levels in SIMOTION, from the task system of the device is integrated in a unit between two labels (start and end labels, compare generic areas for *SetValue*) with this command. Prerequisite is that a unit is already available in the project.

If the information is to be imported into different areas, the parameter line must be called several times.

## Command parameters

| Parameter | |
|---|---|
| TargetName | A reference to a unit or a library is specified here. |
| Unit | If a reference to a library is specified in *TargetName*, the unit of the library is specified here. |
| StartLabel | Start of the area to be replaced. |
| EndLabel | End of the area to be replaced. |
| NameTaskLabel | Label in the code fragment that is replaced by the task name. The label is supplemented by a leading "<" and following ">" character. |
| FileTempl | Specifies the path of the file for the generic code fragment. |

| Generic labels | |
|---|---|
| <counter> | Current index counter |
| <IF_ELSIF> | |

## Example

Table 2- 92 Code example

```
<Command
    ID="100"
    Name="SetTaskInformationInUnit"
    NCID="20">
    <Parameter
       TargetName="fLMsgHdlInit"
       Unit=""
       StartLabel="** start label task info **"
       EndLabel="** end label task info **"
       NameTaskLabel="TaskName"
       FileTempl="SIMOTION\EquipmentModules\V4_3\Messagehandling_LMsgHd
                  l\Data\Templates\TaskInfoblock.st"/>
</Command>
```

The file specified in the FileTempl parameter appears as follows:

```
// <TaskName>
    <IF_ELSIF> (actualTaskId = _task.<TaskName>)
    THEN
        FCLMsgHdlInitTaskInfo  := <counter>;
```

Before the command is called, the area in the imported unit appears as follows:

```
// <<** start label task info **>>

// <<** end label task info **>>
```

Once the command has been called, a section is added for each deselected task:

```
// <<** start label task info **>>

// MotionTask_1
    IF (actualTaskId = _task.MotionTask_1)
    THEN
        FCLMsgHdlInitTaskInfo  := 0;

    // IPOsynchronousTask
    ELSIF (actualTaskId = _task.IPOsynchronousTask)
    THEN
        FCLMsgHdlInitTaskInfo  := 1;

    // ServoSynchronousTask
    ELSIF (actualTaskId = _task.ServoSynchronousTask)
    THEN
        FCLMsgHdlInitTaskInfo  := 2;

    // IPOsynchronousTask_2
    ELSIF (actualTaskId = _task.IPOsynchronousTask_2)
    THEN
        FCLMsgHdlInitTaskInfo  := 3;

    // DccAux
    ELSIF (actualTaskId = _task.DccAux)
        FCLMsgHdlInitTaskInfo  := 4;

    // DccAux_2
    ELSIF (actualTaskId = _task.DccAux_2)
    THEN
        FCLMsgHdlInitTaskInfo  := 5;

...
// <<** end label task info **>>
```

### 2.5.1.4 SetValue

Two basic functions are implemented with this command:

- You can set ST variables and constants to a defined value that is also partially determined during runtime.
  The parameters *Value*, *Name*, *TargetName*, *Type* and *Unit* of the command are used.
- You can set areas in which different code fragments are generated depending on the number of fragments and used labels, and can then be inserted between area tags.
  All parameters are active for the generic areas. A collection of a control (e.g. *Listbox*.SelectedItems) can be transferred on the *Value* parameter with the prefix *__Ref_*. This identifies all the elements of the *collection* as generic area elements.

As precondition for this command, an *ImportUnit* or *ImportLibrary* must be executed.

#### Command parameters

| Parameter | |
|---|---|
| Value | The value of the ST variable or the ST constant is specified here.<br>The value can also be assigned a reference to a user interface element. The text of the user interface element is read out here.<br>The value can be determined during runtime via the prefix *__call_* and a generic constant if the precise value is not defined yet for the configuration. |
| Name | The name of the ST variable or the constant is entered here. Only the first position that was found in the code, is replaced. |
| TargetName | The library is specified here for the type *Library_Constant*.<br>The unit is specified here for the type *Unit_Constant*. |
| Type | The type of the ST variable or the constant is set here:<br>*Library_Constant* or *Unit_Constant* |
| Unit | The target unit is set here when the type *Library_Constant* is selected. Otherwise no meaning. |
| StartLabel | The start label for the area must be specified here.<br>The label is supplemented by leading "<<" and following ">>" characters. |
| EndLabel | The end label for the area must be specified here.<br>The label is supplemented by leading "<<" and following ">>" characters. |
| ArrayLabel | The label that is always replaced by the generic element in the code fragment must be specified here. |
| TemplPath | The path to the code fragment that is replaced between the start and end labels (depending on the number of elements) must be specified here. |
| DisableAreaCheck | Optional parameter to switch off checking whether an area is overwritten between the StartLabel and the EndLabel.<br>Default is FALSE: Test switched on<br>TRUE: Test switched off |
| IndexConstant | Optional: If an area with code fragments is created by SetValue, their number can be assigned a constant in the ST source file. The name of the constant must be specified for this parameter. |

AllowZeroValue Optional: If this parameter is set to TRUE, then 0 or an empty string can be set for the value.

**Element subparameter**

Value The generic area elements are entered here.

Generic constants that are called via a __CALL_<name of the constant>:

- MAXNUMBER_OF_AXES
- MAXNUMBER_OF_CAMS
- MAXNUMBER_OF_OUTPUTCAMTRACKS
- MAXNUMBER_OF_CONTROLLERS
- MAXNUMBER_OF_EXTERNALENCODERS
- MAXNUMBER_OF_FIXEDGEARS
- MAXNUMBER_OF_GEARS
- MAXNUMBER_OF_FORMULAS
- MAXNUMBER_OF_MEASURESWITCHES
- MAXNUMBER_OF_OUTPUTCAMS
- MAXNUMBER_OF_PATH_OBJECTS
- MAXNUMBER_OF_SENSORS
- MAXNUMBER_OF_TEMPERATURE_CONTROLLERS
- MAXNUMBER_OF_ADDERS
- MAXNUMBER_OF_TOS_WITH_DO
- MAXNUMBER_OF_SYSTEM_TASKS
- MAXNUMBER_OF_CYCLIC_DOS
- MAXNUMBER_OF_NON_CYCLIC_DOS
- MAXNUMBER_OF_PERIPHERAL_DEVICES
- INCREMENT_VALUE (value)

## Example

Table 2- 93 Code example

```
<!--Set the Constant LMSGHDL_NUMBER_OF_AXES to the value
    of the generic constant MAXNUMBER_OF_AXES   -->
<Command
    ID="80"
    Name="SetValue"
    NCID="5">
    <Parameter
        Value="__CALL_MAXNUMBER_OF_AXES"
        Name="LMSGHDL_NUMBER_OF_AXES"
        TargetName="LMsgHdl"
        Type="Library_Constant"
        Unit="cPublic"/>

<!—-insert the code from the file AxisReference.st between the start
     and end label and replace the label TO_Name with the names of the
     selected axes in the listbox LB_Axes   -->

    <Parameter
        Value="__ref_LB_Axes.SelectedItems"
        TargetName="xPTemplAxisFB"
        Type="Unit_Constant"
        Unit=""
        StartLabel="** start label axes references axis fb**"
        EndLabel="** end label axes references axis fb **"
        ArrayLabel="TO_Name"
        TemplPath="SIMOTION\EquipmentModules\V4_3\AxisFB_LMCBasic\
                   Data\Templates\AxisReference.st" />
</Command>
```

The AxisReference.st file contains the following code sequence:

```
gasHMIdata[<counter>].sAxisData.toAxis  := _to.<TO_Name>;
gasHMIdata[<counter>].sgTOName          := '<TO_Name>';
```

Assuming that two axes with the names Axis1 and Axis2 are selected in the LB_Axes list box, the following source code is inserted between the start and end label once the command has been called:

```
gasHMIdata[0].sAxisData.toAxis  := _to.Axis1;
gasHMIdata[0].sgTOName          := 'Axis1';
gasHMIdata[1].sAxisData.toAxis  := _to.Axis2;
gasHMIdata[1].sgTOName          := 'Axis2';
```

## 2.5.2 Commands for the object manipulation

### 2.5.2.1 RenameFOO

The following object (FOO) of a following axis is renamed with this command.

As precondition for this command, an *ImportTO* must be executed.

#### Command parameters

| Parameter | |
|---|---|
| TO | Name of the following axis |
| Name | New name of the FOO |
| Index | Each individual FOO can be addressed for renaming via the parameter. If no *Index* is specified, the FOO with index 0 is always renamed. |

#### Example

Table 2- 94 Code example

```
<Command
    ID="100"
    Name="RenameFOO"
    NCID="20">
    <Parameter
        TO="Axis_1"
        Name="FOO_Axis_1"/>
    <Parameter
        TO="Axis_2"
        Name="FOO_Axis_2"
        Index="1"/>
</Command>
```

### 2.5.2.2 SetDeviceSystemVariable

A system variable is set in the active device with this command.

## Command parameters

| Parameter | |
|---|---|
| Name | Complete name of the variable |
| Value | Value that is to be set |

## Example

Table 2- 95 Code example

```
<Command
    ID="100"
    Name="SetDeviceSystemVariable"
    NCID="20">
    <Parameter
        Name="_startupData.operationMode"
        Value="RUN"/>
</Command>
```

### 2.5.2.3 SetMasterTO

The synchronous operation connection between an axis and a following axis is activated with this command.

## Command parameters

| Parameter | |
|---|---|
| FollowingAxis | Name of the slave TO |
| LeadingObject | Name of the master TO |

## Example

Table 2- 96 Code example

```
<Command
    ID="100"
    Name="SetMasterTO"
    NCID="20">
    <Parameter
        FollowingAxis="CB_TO_reference_Belt_A"
        LeadingObject="CB_TO_reference_Belt_B"/>
    <Parameter
        FollowingAxis="CB_TO_reference_Belt_B"
        LeadingObject="CB_TO_reference_Belt_A"/>
</Command>
```

### 2.5.2.4 SetProgram

A program of a unit is assigned to the execution system with this command. You can also specify the target position within the task (optional). The existing programs are moved back if there is a duplicate assignment. If several programs of the same unit or different units are entered in the execution system, the parameter line can be created several times in the same command call.

The prerequisite for this command is that the unit has already been imported with the ImportUnit command and that a program with the specified name exists. The name of the tasks must be specified exactly as it appears in SIMOTION SCOUT. If a MotionTask, TimerInterruptTask, or UserInterruptTask is renamed in the project, the new name must be specified.

## Command parameters

| Parameter | |
|---|---|
| Name | Name of the program |
| Unit | Name of the unit |
| TargetTask | Name of the target task (selectable task) |
| Position | Optional: The target position within the task can be specified here. If *First* is specified, the specified program is entered at the first position in the list of programs of this task. |

## Example

Table 2- 97 Code example

```
<Command
    ID="100"
    Name="SetProgram"
    NCID="20">
    <Parameter
        Name="pStartupCheck"
        Unit="pStartupCheck"
        TargetTask="BackgroundTask"
        Position="First"/>
</Command>
```

### 2.5.2.5 SetTOConfigData

A configuration data item of a technology object can be set to a value with this command.

#### Command parameters

| Parameter | |
|---|---|
| TO | Specifies the name of the TO |
| Name | Complete name of the configuration data item |
| Value | Value of the configuration data item to be set |

The parameter line can be used several times.

#### Example

Table 2- 98 Code example

```
<Command
    ID="100"
    Name="SetTOConfigData"
    NCID="20">
    <Parameter
        TO="CB_TO_reference_Belt_A"
        Name="Modulo.length"
        Value="TB_Belt_length"/>
</Command>
```

### 2.5.2.6 SetTOSystemVariable

A system variable of a technology object can be set to a value with this command.

## Command parameters

| Parameter | |
|---|---|
| TO | Specifies the name of the TO |
| Name | Complete name of the system variable |
| Value | Value of the system variable to be set |

## Example

Table 2- 99 Code example

```
<Command ID="72"
    Name="SetTOSystemVariable"
    NCID="69">
    <Parameter TO="CB_TO_reference_BELT_A"
               Name="userdefaultdynamics.direction"
               Value="negative">
    </Parameter>
</Command>
```

### 2.5.2.7 RemoveProgram

A program can be removed from the execution system with this command. If several programs are to be deleted from the execution system, the parameter line can be specified several times.

## Command parameters

| Parameter | |
|---|---|
| Device | Optional: Name of the SIMOTION device. If this parameter is not specified, the command is used on the active device. |
| UnitName | Name of the unit. |
| Program | Name of the program. |
| Task | Name of the task from which the program is to be removed. |

## Example

Table 2- 100 Code example

```
<Command ID="105" Name="RemoveProgram">
     <Parameter
          UnitName="MyUnit"
          Program="MyProgram"
          Task="BackgroundTask"
     />
</Command>
```

### 2.5.2.8 RenameSubObject

A device subobject can be renamed with this command. If several subobjects are to be renamed, the parameter line can be specified several times.

## Command parameters

| Parameter | |
|---|---|
| Device | Name of the device on which the subobject is located. |
| OldName | The previous name of the subobject. |
| NewName | The new name of the subobject. |

## Example

Table 2- 101 Code example

```
<Command ID="200" Name="RenameSubObject">
     <Parameter
          Device="SINAMICS_Integrated"
          OldName="Control_Unit"
          NewName="ControlUnit1"
     />
</Command>
```

### 2.5.2.9 CreateSymbolicAssignmentForIO

The symbolic assignment for an I/O variable can be set up with this command. The symbolic assignment must be activated within the project – see command: *UseSymbolicAssignment*. If the interconnection is to be set up for several I/O variables, the parameter line can be specified several times.

## Command parameters

| Parameter | |
|---|---|
| Device | Name of the device on which the I/O variable is located. |
| I/O | Name of the I/O variable. |
| Assignment | Name of the assignment partner. |

## Example

Table 2- 102 Code example

```
<Command ID="13" Name="CreateSymbolicAssignmentForIO" NCID="14">
     <Parameter
          Device="NewController"
          IO="Value1"
          Assignment="SINAMICS_Integrated.Control_Unit.CU_ZSW1"
     />
</Command>
```

### 2.5.2.10 SetInterconnection

Different TO interconnections can be set with this command. This function can only be used with SIMOTION SCOUT as of Version 4.4.

If several interconnections are to be created, the parameter line can be specified several times.

## Command parameters

| Parameter | |
|---|---|
| SourceDevice | Name of the device for the interconnection on the input side. |
| SourceTO | Name of the technology object for the interconnection on the input side. |
| Type | InterconnectionID of the input interface. |
| MappedDevice | Name of the device for the interconnection on the output side. |
| MappedTO | Name of the technology object for the interconnection on the output side. |
| MappedType | InterconnectionID of the output interface. |

## Example

Table 2- 103 Code example

```
<Command ID="12345" Name="SetInterconnection">
     <Parameter
          SourceDevice="newdevice"
          SourceTO="Axis_1"
          Type="AdditiveTorque"
          MappedDevice="newdevice"
          MappedTO="Axis_2"
          MappedType="ActualTorque"
     />
</Command>
```

### 2.5.2.11 RemoveInterconnection

An interconnection between technology objects can be deleted with this command. This function can only be used with SIMOTION SCOUT as of Version 4.4.

If several interconnections are to be deleted, the parameter line can be specified several times.

## Command parameters

| Parameter | |
|---|---|
| SourceDevice | Name of the device for the interconnection on the input side. |
| SourceTO | Name of the technology object for the interconnection on the input side. |
| Type | InterconnectionID of the input interface. |
| MappedDevice | Name of the device for the interconnection on the output side. |
| MappedTO | Name of the technology object for the interconnection on the output side. |
| MappedType | InterconnectionID of the output interface. |

## Example

Table 2- 104 Code example

```
<Command ID="1111" Name="RemoveInterconnection">
     <Parameter
          SourceDevice="newdevice"
          SourceTO="Axis_1"
          Type="AdditiveTorque"
          MappedDevice="newdevice"
          MappedTO="Axis_2"
          MappedType="ActualTorque"
     />
</Command>
```

### 2.5.2.12 ConfigureProfinetIRT

The settings required for the IRT communication can be made with this command. All IRT-capable devices are set up as sync slave and the controller as sync master in the respective PROFINET IO system.

The times specified in the parameters are set and all IRT-capable slots activated as slaves. The Ti/To mode is set to "IO Device" and the specified times set for all devices.

## Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the controller to be configured as sync master. |
| CPUInterface | Specification of the controller interface on which the subsystem to be configured is located.<br>Possible values:<br>PN1 when using the X150 SIMOTION interface<br>PN2 when using the X1400 SIMOTION interface |
| SyncDomain | Optional: Name of the sync domain. If this parameter is not specified, the standard sync domain is used. |
| CycleTime | Bus cycle time in ms. |
| Ti | Time to read-in the process values in µs. |
| To | Time to output the process values in µs. |

## Example

Table 2- 105 Code example

```
<Command ID="303" Name="ConfigureProfinetIRT">
    <Parameter
        DeviceName="newDevice"
        CPUInterface="PN1"
        SyncDomain=""
        CycleTime ="4000"
        Ti="125"
        To="250"
    />
<Command/>
```

### 2.5.2.13 SetProfibusIntegratedCycleTime

The settings for the isochronous mode of the integrated PROFIBUS and all subordinate devices (SINAMICS–Integrated and CX) are set with this command.

The Ti/To mode is set to "IO Device" and the specified times set for all devices.

## Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the controller. |
| MasterApplicationCycleTime | Master application cycle in µs. |
| CycleTime | Bus cycle time in ms. |
| Ti | Time to read-in the process values in µs. |
| To | Time to output the process values in µs. |

## Example

Table 2- 106 Code example

```
<Command ID="316" Name="SetProfibusIntegratedCycleTime">
   <Parameter
      Device="SINAMICS_Integrated"
      SubnetName="Profibus Integrated"
      MasterApplicationCycleTime ="4000"
      CycleTime ="4000"
      Ti="125"
      To="250"
   />
</Command>
```

### 2.5.2.14 AddPNDevice

Arbitrary PROFINET devices can be inserted in a PROFINET IO system by means of the MLFB or GSDML file with this command.

In special cases, a prefix must be attached to the MLFB. For this reason, it is recommended that an export is created of the relevant station from HW Config and that the MLFB specified there is used.

If a GSDML file is used, it must already be installed in the hardware configuration. If more than one device or module is defined in the GSDML file, the appropriate ID from the GSDML file must be specified for the device.

Optionally, as many modules and submodules as permitted by the engineering system can be inserted in this device. The *Module* and *SubModule* elements can be specified a corresponding number of times.

Limitation of the possible ET200 stations: See supplementary conditions general information number 15.

#### Command parameters

| Parameter | |
|---|---|
| DeviceName | Name to be assigned for the inserted device. |
| DeviceIdentifier | Specification of the MLFB or GSDML file. |
| ID | Optional: Specification of the ID for the specification of the device from the GSDML file. |
| IPAddress | Specification of the IP address (also possible as hex value), e.g.<br>192.168.0.6<br>C0A80006 |
| CPUInterface | Specification of the controller interface on which the subsystem in which the device is to be inserted is located. |
| Version | Optional: Specification of the version if the DeviceIdentifier parameter is an MLFB. |
| DeviceNumber | Optional: Specification of the device number in the PROFINET IO system if a number other than the next free number is desired. |
| **Module** | |
| ModuleName | Name to be assigned for the inserted module. |
| ModuleIdentifier | Specification of the MLFB or GSDML file. |
| ModuleID | Optional: Specification of the ID for the specification of the module from the GSDML file. |
| SlotNumber | Specification of the slot number for the addressing of subordinate slots. |
| **SubModule** | |
| SubModuleName | Name to be assigned for the inserted submodule. |

| | |
|---|---|
| SubModuleIdentifier | Specification of the MLFB or GSDML file. |
| SubModuleID | Optional: Specification of the ID for the specification of the submodule from the GSDML file. |
| SlotNumber | Specification of the slot number for the addressing of subordinate slots. |
| SubSlotNumber | Specification of the subslot number for the addressing of subordinate subslots. |

## Example

Table 2- 107 Code example

```
<Command ID="307" Name="AddPNDevice">
    <Parameter
        DeviceName="__ref_TB_GSDName.Text"
        DeviceIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
        ID="DIM_20" IPAddress="192.168.1.6"
        CPUInterface="PN1" Version="V2.0"
        Address="2">
        <Module
            ModuleName="DI 16x24VDC ST"
            ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            ModuleID="DI 16x24VDC ST" SlotNumber ="1"/>
        <Module ModuleName="DI 16x24VDC ST"
            ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            ModuleID="DI 16x24VDC ST" SlotNumber ="2"/>
        <Module
            ModuleName="DI 16x24VDC ST"
            ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            ModuleID="DI 16x24VDC ST"
            SlotNumber ="3"/>^
        <SubModule
            SubModuleName ="My Port 1 (2xRJ45)"
            SubModuleIdentifier ="GSDML-V2.31-Siemens-ET200SP-
20150326.xml"
            SubModuleID ="IDS_1P1 HF RJ45 V2.2"
            SlotNumber ="0"
            SubSlotNumber ="2"/>
    </Parameter>
</Command>
```

### 2.5.2.15 AddPNDeviceModule

Individual modules can be added to a PROFINET device by means of the GSDML file with this command.

As prerequisite, the GSDML file to be used must already be installed in HW Config.

Any number of modules and submodules can be inserted in a device with a single command.

A module can be created with the *Module* element. Several modules can be created through multiple use of the element.

A submodule can be created with the *SubModule* element. Several submodules can be created through multiple use of the element.

## Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the device in which the modules or submodules are to be inserted. |

| Module | |
|---|---|
| ModuleName | Name to be assigned for the inserted module. |
| ModuleIdentifier | Specification of the MLFB or GSDML file. |
| ModuleID | Specification of the ID for the specification of the module from the GSDML file. |
| SlotNumber | Specification of the slot number for the addressing of subordinate slots. |

| SubModule | |
|---|---|
| SubModuleName | Name to be assigned for the inserted submodule. |
| SubModuleIdentifier | Specification of the MLFB or GSDML file. |
| SubModuleID | Optional: Specification of the ID for the specification of the submodule from the GSDML file. |
| SlotNumber | Specification of the slot number for the addressing of subordinate slots. |
| SubSlotNumber | Specification of the subslot number for the addressing of subordinate subslots. |

### Example

Table 2- 108 Code example

```
<Command ID="310" Name="AddPNDeviceModule">
    <Parameter
        DeviceName ="__ref_TB_GSDName.Text">
        <Module
            ModuleName="DI 16x24VDC ST"
            ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            ModuleID="DI 16x24VDC ST"
            SlotNumber ="1"/>
        <SubModule
            SubModuleName ="My Port 1 (2xRJ45)"
            SubModuleIdentifier ="GSDML-V2.31-Siemens-ET200SP-
20150326.xml"
            SubModuleID ="IDS_1P1 HF RJ45 V2.2"
            SlotNumber ="0"
            SubSlotNumber ="2"/>
    </Parameter>
</Command>
```

## 2.5.2.16 CreateET200Module

Individual modules of an ET200 can be created and configured with this command.

### Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the ET200 station. |
| ModuleName | Name of the module. |
| MLFB | MLFB of the module. |
| SlotNumber | Slot number in which the module is to be inserted. |
| | |
| InAddress | Optional: Start address of the module. |
| InAddressSMType | Optional: Is only used for ET 200L and ET 200S modules and has the following coding:<br>16: Digital addresses<br>32: Analog addresses<br>256: Diagnostic addresses |
| InAddressLength | Optional: |
| InSubAddress | Optional: Subaddress of the module. |
| InPartProcessImage | Optional: Number of the assigned part process image. |

| | |
|---|---|
| InArea | Optional: Area identifier.<br>0: SIMATIC 400 module<br>1: SIMATIC 300 module<br>2: DP addressing<br>3: P area (S5 connection)<br>4: Q area (S5 connection)<br>5: IM3 area (S5 connection)<br>6: IM4 area (S5 connection)<br>7: Reserved |
| InBitAddress | Optional: The bit offset to the start address is stored here for ET 200 and ET 200S electronic modules. |
| OutAddress | Optional: Specification of the logical address. |
| OutAddressSMType | Optional: Is only used for ET 200L and ET 200S modules and has the following coding:<br>16: Digital addresses<br>32: Analog addresses<br>256: Diagnostic addresses |
| OutAddressLength | Optional: Length of the address in bytes (depending on the module) |
| OutSubAddress | Optional: Subaddress of the module. |
| OutPartProcessImage | Optional: Number of the assigned part process image. |
| OutArea | Optional: Area identifier.<br>0: SIMATIC 400 module<br>1: SIMATIC 300 module<br>2: DP addressing<br>3: P area (S5 connection)<br>4: Q area (S5 connection)<br>5: IM3 area (S5 connection)<br>6: IM4 area (S5 connection)<br>7: Reserved |
| OutBitAddress | Optional: The bit offset to the start address is stored here for ET 200 and ET 200S electronic modules. |

## Example

Table 2- 109 Code example

```
<Command ID="321" Name="CreateEt200Module">
    <Parameter ET200Name="MyET200"
        ModuleName="TestxCreateEt200Module"
 MLFB="6ES7 131-4BF50-0AA0" Version="V7.0"
        SlotNumber="4"
        InAddress="42"
        InAddressSMType=""
        InAddressLength=""
        InSubAddress=""
        InPartProcessImage=""
        InArea=""
        InBitAddress=""
        OutAddress=""
        OutAddressSMType=""
        OutAddressLength=""
        OutSubAddress=""
        OutPartProcessImage=""
        OutArea=""
        OutBitAddress=""
    />
</Command>
```

### 2.5.2.17 SetDistributedSynchronousOperation

The distributed synchronous operation can be configured through controller-controller direct communication with this command. The following object is connected to the master object. The PROFINET connection of the two controllers must be set up first (see CreatePNTopology). The controller must also be sync master with the master object and the other controller sync slave. This can be set, for example, with SetDeviceProperty.

## Command parameters

| Parameter | |
|---|---|
| MasterObjectDevice | Name of the CPU with the master object. |
| FollowingObjectDevice | Name of the CPU with the following object. |
| MasterObject | Name of the master object. |
| FollowingObject | Name of the following object, not the name of the following axis. |
| MasterInterconnectionID | Optional: InterconnectionID of the leading axis per default assigned "MotionCommandValue". |
| FollowingInterconnectionID | Optional: InterconnectionID of the following axis per default assigned "MotionCommandValue". |

## Example

Table 2- 110 Code example

```
<Command ID="311" Name="SetDistributedSynchronousOperation">
    <Parameter
        MasterObjectDevice ="newDevice"
        FollowingObjectDevice ="newDevice1"
        MasterObject ="TOxMaster"
        FollowingObject ="TOxSlave_FOO1"
        MasterInterconnectionID ="MotionCommandValue"
        FollowingInterconnectionID ="MotionCommandValue"
    />
</Command>
```

### 2.5.2.18 CreatePNTopology

The PROFINET topology can be configured with this command. The ports are connected according to the parameter specifications. Whereby, the Device as well as the PartnerDevice can be a station or a slave.

The Parameter element can be used several times for multiple interconnections (see example).

## Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the controller/device. |
| PNInterfaceName | Name of the PROFINET interface. |
| PortNumber | Number of the port. |
| PartnerDeviceName | Name of the partner controller / device. |
| PartnerPNInterfaceName | Name of the partner PROFINET interface. |
| PartnerPortNumber | Number of the partner port. |

## Example

Table 2- 111 Code example

```
<Command ID="305" Name="CreatePNTopology">
    <Parameter
        DeviceName ="newDevice"
        PNInterfaceName ="PNxIO"
        PortNumber ="3"
        PartnerDeviceName ="NewCU"
        PartnerPNInterfaceName ="PN-IO"
        PartnerPortNumber ="1"
    />
    <Parameter
        DeviceName ="newDevice"
        PNInterfaceName ="PNxIO"
        PortNumber ="2"
        PartnerDeviceName ="GSDxDevice"
        PartnerPNInterfaceName ="Interface"
        PartnerPortNumber ="1"
    />
</Command>
```

### 2.5.2.19 CreateProfinetSubsystem

PROFINET IO systems can be created at the specified interface of the active controller with this command.

If a subnet with the specified name already exists, it is linked to the PROFINET IO system. Otherwise a new subnet is created and linked.

The name of the new subsystem is "PROFINET IO System" and is assigned the next free IO system number.

## Command parameters

| Parameter | |
|---|---|
| CPUInterface | Specification of the controller interface on which the subsystem in which the device is to be inserted is located.<br>Possible values:<br>PN1 when using the X150 SIMOTION interface<br>PN2 when using the X1400 SIMOTION interface |
| SubnetName | Name of the subnet. |

## Example

Table 2- 112 Code example

```
<Command ID="319" Name="CreateProfinetSubsystem">
    <Parameter
        SubnetName ="Ethernet(101)"
        CPUInterface ="PN1"
    />
</Command>
```

# 2.6 Making settings for technology packages

## 2.6.1 SetLibraryTP

The technology package of a library to be imported can be changed with this command. Either just one device with the associated technology package can be identified, or an automatic identification started across the entire project.

Prerequisite for this command is that a library is already available in the project.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the library |
| Version | The version of the associated device must be specified here if no automatic identification is activated.<br>Examples: V4_2, V4_3, V4_4 |
| Device | Specifies the device type if no automatic identification is activated. The device type complies with the file name under "SIMOTION\TechnologyPackages\[the respective version]\DeviceType without ".txt" (e.g. D425-2). If "All" is specified for this parameter, all known device types are used. |
| TP | The automatic identification is activated with the value *automatic*. Otherwise the type of the technology package can be specified, e.g. TPCAM, TPCAMEXT, TPPATH. |

### Example

Table 2- 113 Code example

```
<Command
    ID="51"
    Name="SetLibraryTP"
    NCID="52">
    <Parameter
        Name="LMsgHdl"
        Version=""
        Device=""
        TP="automatic"/>
</Command>
```

## 2.6.2 SetUsepackage

All technology objects of the active device are searched and the required technology package set within the unit with this command.

The parameter line can be called several times.

### Command parameters

| Parameter | |
|---|---|
| Unit | Specifies the target unit |

### Example

Table 2- 114 Code example

```
<Command
    ID="100"
    Name="SetUsepackage"
    NCID="20">
    <Parameter
        Unit="pStartupCheck"/>
</Command>
```

# 2.7 Commands for the define handling

## 2.7.1 DeleteDefine

The defines on an existing source (unit or library unit) can be deleted with this command.
Several sources can be addressed with one system call if the parameter line is integrated several times.
Several defines of a source can be deleted if the subelement *Define* below the parameter line is called several times.

### Command parameters

| | |
|---|---|
| **Parameter** | |
| Library | Name of the target library. If no library unit is used, this parameter is omitted. |
| Unit | Name of the unit |
| **Define** | |
| Name | Name of the define to be deleted |

## Example

Table 2- 115 Code example

```
<Command
    ID="100"
    Name="DeleteDefine"
    NCID="20">
    <Parameter
        Library="LMsgHdl"
        Unit="cPublic">
        <Define
            Name="LMSGHDL_TP_CAM" />
        <Define
            Name="LMSGHDL_TP_PATH" />
        <Define
            Name="LMSGHDL_TP_CAM_EXT" />
        <Define
            Name="LMSGHDL_TP_TCONTROL"/>
    </Parameter>
    <Parameter
        Library=""
        Unit="fLMsgHdlInit">
        <Define
            Name="LMSGHDL_TP_CAM" />
        <Define
            Name="LMSGHDL_TP_PATH" />
        <Define
            Name="LMSGHDL_TP_CAM_EXT" />
        <Define
            Name="LMSGHDL_TP_TCONTROL"/>
        <Define
            Name="LMSGHDL_NO_DO_MESSAGES"/>
        <Define
            Name="LMSGHDL_NO_TIME_SYNC"/>
        <Define
            Name="LMSGHDL_NO_STRING_MESSAGES"/>
        <Define
            Name="LMSGHDL_NO_SIMOTION_IT_HMI"/>
    </Parameter>
</Command>
```

## 2.7.2 RestoreDefines

Defines of a unit that are set in the properties of the source at Further Settings – Preprocessor Definitions, are restored with this command when the unit or library unit is already present in the project and is to be replaced with a new one. As precondition for this command, either an *ImportUnit* or an *ImportLibrary* must be executed.

### Command parameters

| Parameter | |
|---|---|
| Library | Specifies the library in which the defines are to be restored. If the command is to apply to a unit below a device, the parameter is omitted. |
| Unit | Specifies the unit in which the defines are to be restored. |

### Example

Table 2- 116 Code example

```
<Command
    ID="100"
    Name="RestoreDefines"
    NCID="20">
    <Parameter
        Library=""
        Unit="pStartupCheck"/>
</Command>
```

## 2.7.3 SetAutoTPDefine

A define can be set automatically in a unit or library unit that depends on the currently required technology package with this command. The define that is set is transferred during the function call.

Prerequisite for this command is that a library or a unit is already available in the project.

### Command parameters

| Parameter | |
|---|---|
| Library | Name of the library; can be omitted if no library unit is to be addressed. |
| Unit | Name of the unit |
| TPCAM | Transfer of the define that is to be set when the CAM technology package is active. |
| TPPATH | Transfer of the define that is to be set when the PATH technology package is active. |
| TPCAMEXT | Transfer of the define that is to be set when the CAMEXT technology package is active. |
| TPTCONTROL | Transfer of the define that is to be set when the TCONTROL technology package is active. |

### Example

Table 2- 117 Code example

```
<Command
    ID="100"
    Name="SetAutoTPDefine"
    NCID="20">
    <Parameter
        Library="LMsgHdl"
        Unit="cPublic"
        TPCAM="LMSGHDL_TP_CAM"
        TPPATH="LMSGHDL_TP_PATH"
        TPCAMEXT="LMSGHDL_TP_CAM_EXT"
        TPTCONTROL="LMSGHDL_TP_TCONTROL"/>
    <Parameter
        Library=""
        Unit="fLMsgHdlInit"
        TPCAM="LMSGHDL_TP_CAM"
        TPPATH="LMSGHDL_TP_PATH"
        TPCAMEXT="LMSGHDL_TP_CAM_EXT"
        TPTCONTROL="LMSGHDL_TP_TCONTROL"/>
</Command>
```

## 2.7.4 SetDefine

A define is set for the SIMOTION compiler with this command.

As precondition for this command, an *ImportUnit* or *ImportLibrary* must be executed.

### Command parameters

| Parameter | |
|---|---|
| Library | Specifies the target library. If the target unit is not a library unit, this parameter can be ignored. |
| Unit | Specifies the target unit. |
| Define | Transfer of the define to be set. |

### Example

Table 2- 118 Code example

```
<Command
    ID="100"
    Name="SetDefine"
    NCID="20">
    <Parameter
        Library=""
        Unit="fLMsgHdlInit"
        Define="LMSGHDL_NO_STRING_MESSAGES"/>
</Command>
```

# 2.8 Commands for the update

## 2.8.1 SetRestoreArea

If a unit available in the project which has the attribute ForceImport = True is imported, it is overwritten. A complete area between the two labels of a unit or library unit is saved so that it is not overwritten.

As a precondition for this command, either an *ImportUnit* or an *ImportLibrary* must be executed.

### Command parameters

| Parameter | |
|---|---|
| Library | Specifies the target library. If the target unit is not a library unit, this parameter can be ignored. |
| Unit | Specifies the target unit. |
| StartLabel | Start of the area to be replaced. |
| EndLabel | End of the area to be replaced. |

### Example

The "Unit with a rescue area" code example demonstrates how an area is defined in the unit in question. If the unit is imported again and the SetRestoreArea command is called (see the "Calling SetRestoreArea" code example), this area remains unchanged. Everything else is

Table 2- 119 Unit with a rescue area code example

```
// RESCUE AREA - area will be restored by script controlled unit exchange
// <<** start label rescue area **>>
   LMSGHDL_NO_MACHINE_ERROR_CLASS                  : SINT :=-1;
   LMSGHDL_MACHINE_ERROR_CLASS0                    : SINT :=0;
   LMSGHDL_MACHINE_ERROR_CLASS1                    : SINT :=1;
   LMSGHDL_MACHINE_ERROR_CLASS2                    : SINT :=2;
   LMSGHDL_MACHINE_ERROR_CLASS3                    : SINT :=3;
   LMSGHDL_MACHINE_ERROR_CLASS4                    : SINT :=4;
// <<** end label rescue area **>>
// END RESCUE AREA
```

Table 2- 120 Calling SetRestoreArea code example

```
<Command ID="100" Name="SetRestoreArea" NCID="20">
    <Parameter
        Library=""
        Unit="fLMsgHdlInit"
        StartLabel="** start label rescue area **"
        EndLabel="** end label rescue area **" />
     <!– Another area in the same unit-->
     <Parameter
        Library=""
        Unit="fLMsgHdlInit"
        StartLabel="** start label rescue area 2 **"
        EndLabel="** end label rescue area 2 **" />
     <!– Area in a library unit-->
     <Parameter
        Library="LMsgHdl"
        Unit="cPublic"
        StartLabel="** start label restore area **"
        EndLabel="** end label restore area **" />
</Command>
```

## 2.8.2 RestoreConstants

Constants of a unit are saved with this command when the unit or library unit is already available in the project.

As precondition for this command, either an *ImportUnit* or an *ImportLibrary* must be executed.

### Command parameters

| Parameter | |
|---|---|
| Library | Specifies the library in which the constants are to be saved. If the target unit is not a library unit, this parameter can be ignored. |
| Unit | Specifies the unit in which the constants are to be saved. |

### Example

Table 2- 121 Code example

```
<Command
    ID="100"
    Name="RestoreConstants"
    NCID="20">
    <Parameter
        Library="LDPV1"
        Unit="cPublic"/>
    <Parameter
        Library="LMsgHdl"
        Unit="cPublic"/>
</Command>
```

# Commands for creating a SINAMICS device 3

## 3.1 Overview of parameters

The table below provides an overview of the most important parameters that can be set for SINAMICS commands in the ProjectGenerator. The parameters are listed together with their meaning, the data type, their possible values, and an example.

Table 3- 1 Overview of the commonly used parameters

| Parameter | Meaning | Data type | Possible values | Code example |
|---|---|---|---|---|
| CPUInterface | Specification of the interface of the active device on whose subsystem the SINAMICS device is to be imported. | ENUM | DP1<br>DP2/MPI<br>DP_INT<br>PN1 | `Source="__call_GetAvailableSinamicsVersionsForInterfaces('DP1')"` |
| DeviceName | Transfer of the SINAMICS device name | STRING | --- | `Source="__call_GetNotConnectedDOs(__ref_Cb_SinamcisStationsOfDevice.Text)"` |

## 3.2 Commands for inserting and deleting objects

### 3.2.1 ImportSinamicsStation

With this command, SINAMICS devices can be imported to a bus system of the active device.

Any number of SINAMICS devices can be imported with one call of the command.
The parameter line must be filled in several times in this case.

#### Command parameters

| Parameter | |
|---|---|
| Name | Name of the SINAMICS devices<br>The value can also be assigned a reference to a user interface element. The *Text* property of the element is evaluated here. |
| CPUInterface | Specification of the interface of the active device on whose subsystem the SINAMICS device is to be imported. |
| Version | Specification of the firmware version with which the SINAMICS device is to be created.<br>If the integrated PROFIBUS is selected at the *CPUInterface*, it is not necessary to specify a version, as it is determined automatically. |

| | |
|---|---|
| FilePath | Path to the export file<br>The path can also be specified relatively.<br>In the parameter "UseLocalPath = True" the path is preallocated with the value "\SINAMICS\Devices\<NameOfExportFile>.XML".<br>The name of the XML file is then given in this parameter.<br>The export must be a SINAMICS device export including all drive objects (DO). |
| Address | If a PROFIBUS interface is selected in the *CPUInterface* parameter, the PROFIBUS address is specified here. If no address is selected, the ProjectGenerator takes the next free address.<br>If a PROFINET interface is selected, this parameter is not relevant.<br>A valid number must be transferred for the integrated PROFIBUS variant, as this cannot be changed later. |
| IP_Address | If a PROFINET interface is selected for the *CPUInterface* parameter, this parameter represents the IP address of the SINAMICS device. |
| UseLocalPath | See parameter *FilePath* |

## Example

Table 3- 2 Code example

```
<Command
    ID="10"
    Name="ImportSinamicsStation"
    NCID="20">
    <Parameter
        Name="__ref_TB_SINAMICSName.Text"
        CPUInterface ="__ref_CB_Com_Interfaces.Text"
        Version ="__ref_CB_Available_Versions.Text"
        FilePath ="__ref_CB_AvailableSinamicsExports.Text"
        Address ="NEXT_FREE"
        IP_Address ="__ref_TB_IP_Address.Text"
        UseLocalPath ="True"/>
</Command>
```

## See also

GetSinamicsStationsOfDevice (Page 166)

## 3.2.2 SetTODOConnection

This command allows a technology object (TO) of the active device to automatically create connections to a drive object (DO) of a subordinate SINAMICS device.

Any number of connections can be created with one call of the command.
The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| TO | Name of the TO to be connected |
| SINAMICS_Device_Name | Name of the SINAMICS device on which the DO to be connected is located |
| DO | Name of the DO to be connected |

### Example

Table 3- 3 Code example

```
<Command
    ID="5040"
    Name="SetTODOConnection"
    NCID="5050">
    <Parameter
        TO="__ref_LB_TOs_in_Device.SelectedItem"
        DO="__ref_LB_DOs_in_SinamicsDevice.SelectedItem"
        SINAMICS_Device_Name="Cb_SinamcisStationsOfDevice"/>
</Command>
```

### See also

GetTODOConnections (Page 167)

### 3.2.3 RemoveTODOConnection

This command allows connections between a technology object (TO) of the active device and a drive object (DO) of a subordinate SINAMICS device to be disconnected.

Only new connections can be deleted.

Any number of connections can be disconnected with one call of the command. The parameter line must be filled in several times in this case.

#### Command parameters

| Parameter | |
|---|---|
| TO | Name of the TO whose connection to the assigned DO should be deleted. A connection may be ended only when there are new connections. |

#### Example

Table 3- 4 Code example

```
<Command
    ID="5060"
    Name="RemoveTODOConnection"
    NCID="5050">
    <Parameter
        TO="__ref_LV_Connected_TO_DOs.SelectedItems(0).SubItems(0).Text"/>
</Command>
```

## 3.2.4 ImportDOToDevice

Drive objects (DO) can be imported into a SINAMICS device with this command. As a precondition for this command, a *SetSinamicsDeviceSettings* must be executed.

Any number of DOs can be imported with one call of the command. The parameter line must be filled in several times in this case.

**Note**

**The DO exports must have a Single Motor Module. This can be replaced by generating with the ProjectGenerator.**

**Note**

**There are three options for creating HW Config slots:**

1. Specification of the target address/distance from the next slot directly on the command.
2. Insertion of the file IOConfig.xml in the first lower level of the export.
   – XML configuration:
   ```
   <IOConfig StartAddress="256"
    DefaultLength="20" />
   ```
3. If neither point 1 or point 2 are fulfilled, the ProjectGenerator starts with the address 256 and a default length of 20.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the DOs to be imported. |
| DeviceName | Name of the SINAMICS device to which the DO is to be imported. |
| PowerUnitOrderNo | Optional: Specification of the target power unit MLFB which is to be set to this DO after import. The X1 connection is used with a Double Motor Module. |
| Ref_DO_Name | Optional: Specification of the DOs with a Double Motor Module whose X2 connection should be used for this DO.<br>The parameter PowerUnitOrderNo must be kept free with this functionality. If the two parameters *PowerUnitOrderNo* and *Ref_DO_Name* remain like this, the assignment of the X2 DOs to the X1 DOs can be generated later, using separate commands. |
| FilePath | Optional: Specification of the path to the XML export that should be imported |
| Line | Optional: Specification of the DRIVE CLiQ socket to the CU. If this parameter is not supplied, standard wiring is installed.<br>• LineModules = Port 0<br>• MotorModules = Port 1<br>• Rest = Port 2<br>• TM45F = Port 3 |

| | |
|---|---|
| ExportName | Optional: Using the parameters *ExportName* and *ExportPath,* the FilePath can be created using 2 parameters.<br>For this purpose, the *ExportPath* can be permanently stored and the *ExportName* can be selected on the user interface through an operator action, for example. |
| ExportPath | Optional: See *ExportName*. |
| StartAddress | Optional: Specification of the HW Config start address for this DO. |
| DefaultLength | Optional: Specification of the distance of the HW Config addresses with the same start address.<br>**Example**:<br>DO1 address = 256 distance 20<br>DO2 address =256 distance 20<br>The result of this through the ProjectGenerator:<br>DO1 = 256<br>DO2 = 276 |

## Example

Table 3- 5 Code example

```
<Command
      ID="1503"
      Name="ImportDOToDevice" >
      <Parameter
            Name="DO_Motor_01"
            DeviceName="CX_1"
            FilePath="SINAMICS\DOs\Drive_1.xml"
            PowerUnitOrderNo="6SL3120-2TE13-0Axx"
            Ref_DO_Name=""
            Line="1"/>
      <Parameter
            Name="DO_Motor_02"
            DeviceName="CX_1"
            FilePath="SINAMICS\DOs\Drive_2.xml"
            PowerUnitOrderNo=""
            Ref_DO_Name="DO_Motor_01"/>
</Command>
```

## 3.2.5 SetSinamicsParameter

Parameters can be set to available SINAMICS objects with this command.

Any number of SINAMICS parameters can be changed with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the SINAMICS object. |
| Number | Number of the parameter, e.g., P1821[0]. |
| Value | Parameter value. |
| DeviceName | Name of the SINAMICS device. |

### Example

Table 3- 6 Code example

```
<Command
      ID="1500"
      Name="SetSinamicsParameter"
      NCID="1502">
      <Parameter
            Name="DO_Motor_01"
            DeviceName="CX_1"
            Number="P1821[0]"
            Value="0"/>
</Command>
```

## 3.2.6 SetSinamicsBiCoConnection

Parameters can be set on available SINAMICS objects with this command.

Any number of SINAMICS parameters can be changed with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the SINAMICS object. |
| Number | Number of the parameter, e.g. P1821. |
| Index | Index of the parameter. |
| SrcName | Name of the source object. |
| SrcNumber | Name of the source parameter. |
| SrcIndex | Index of the source parameter. |
| DeviceName | Name of the SINAMICS device. |

### Example

Table 3- 7 Code example

```
<Command
      ID="1500"
      Name="SetSinamicsBiCoConnection"
      NCID="1502">
      <Parameter
            DeviceName="CX_1"
            Name="DO_Motor_01"
            Number="P864"
            Index=""
            SrcName="Control_Unit"
            SrcNumber="8510"
            SrcIndex="0"/>
</Command>
```

## 3.2.7 SetDOToDMMSlot

DOs can be connected to a free X2 connection of a DO with Double Motor Module with this command. As a precondition for this command, a *SetSinamicsDeviceSettings* must be executed and the DOs must be imported via *ImportDOToDevice.*

Any number of connections can be established with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the DOs that should be connected to the X2 of a Double Motor Module. |
| Ref_DO_Name | Name of the DO that should be connected to a free X2 of a Double Motor Module. |
| SINAMICS_Device_Name | Name of the SINAMICS device on which the two DOs are located. |

### Example

Table 3- 8 Code example

```
<Command
      ID="5040"
      Name="SetDOToDMMSlot"
      NCID="5050">
      <Parameter
            Name="__ref_LB_TOs_in_Slave.SelectedItem"
            Ref_DO_Name="__ref_LB_DOs_in_Slave.SelectedItem"

SINAMICS_Device_Name="__ref_Cb_SinamcisStationsOfDevice.Text"/>
</Command>
```

## 3.2.8 DeleteDOFromDMMSlot

The connections that were created with the command *SetDOToDMMSlot* can be disconnected again with this command.

Any number of connections can be disconnected with one call of the command. The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the DO that should be disconnected from the X2 of a Double Motor Module. |
| Ref_DO_Name | Name of the DO which is itself on X1. |
| SINAMICS_Device_Name | Name of the SINAMICS device on which the two DOs are located. |

### Example

Table 3- 9 Code example

```
<Command
      ID="5060"
      Name="DeleteDOFromDMMSlot"
      NCID="5050">
<Parameter
      Name="__ref_LV_Connected_TO_DOs.SelectedItems(0).SubItems(0).Text"

Ref_DO_Name="__ref_LV_Connected_TO_DOs.SelectedItems(0).SubItems(1).Text"
      SINAMICS_Device_Name="__ref_Cb_SinamcisStationsOfDevice.Text"/>
</Command>
```

## 3.2.9 SetSinamicsDeviceSettings

With this command, SINAMICS devices can be created on a bus system of the active device. If a SINAMICS device already exists, (e.g. SINAMICS_Integrated), only *CPUInterface* and *Name* of the device is necessary.

Any number of SINAMICS devices can be inserted with one call of the command.

The parameter line must be filled in several times in this case.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the SINAMICS station. |

| | |
|---|---|
| CPUInterface | Specification of the interface of the active device on whose subsystem the SINAMICS device is to be imported. |
| Version | Specification of the firmware version with which the SINAMICS device is to be created. |
| Address | If a PROFIBUS interface is selected in the *CPUInterface* parameter, the PROFIBUS address is specified here. If no address is selected, the ProjectGenerator takes the next free address.<br>If a PROFINET interface is selected, this parameter is not relevant.<br>A valid number must be transferred for the integrated PROFIBUS variant, as this cannot be changed later. |
| ExternalTypeName | Specification whether SINAMICS_CU320 or SINAMICS_S120_CX32. |
| IP_Address | If a PROFINET interface is selected for the *CPUInterface* parameter, this parameter represents the IP address of the SINAMICS device. |
| CUTelegram | Optional: Specifies the message frame to be set for the CU. |
| StartAddress | Optional: Specification of the HW Config start address for this DO. |
| DefaultLength | Optional: Specification of the distance of the HW Config addresses with the same start address.<br>**Example**:<br>DO1 address = 256 distance 20<br>DO2 address =256 distance 20<br>The result of this through the ProjectGenerator:<br>DO1 = 256<br>DO2 = 276 |

## Example

Table 3- 10 Code example

```
<Command
      ID="12"
      Name="SetSinamicsDeviceSettings">
      <Parameter
            Name="SINAMICS_Integrated"
            CPUInteface="DP_INT"/>
</Command>
```

## 3.2.10 RenameSinamicsIntegrated

The SINAMICS Integrated of a SIMOTION device can be renamed with this command. If this command is used during the generation of a new project, a *SetSinamicsDeviceSettings* for the SINAMICS Integrated must be executed as precondition for this command.

If several SIMOTION devices are imported, note that SIMOTION SCOUT changes the names by adding a suffix if there are SINAMICS Integrated devices with the same name. This command is used on the active SIMOTION device.

### Command parameters

| Parameter | |
|---|---|
| OldName | Previous name of the SINAMICS Integrated. |
| NewName | New name of the SINAMICS Integrated. |

### Example

Table 3- 11 Code example

```
<Command ID="311" Name="RenameSinamicsIntegrated">
     <Parameter
          OldName="SINAMICS_Integrated"
          NewName="SI_D435"
     />
</Command>
```

# 3.3 Information commands

## 3.3.1 GetAvailableInterfacesOfDevice

This command reads out and returns all available interfaces of a SIMOTION device.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

This command does not have any parameters.

### Example

Table 3- 12 Code example

**Transfer to a control**

```
Source="__call_GetAvailableInterfacesOfDevice"
```

**Use in the source code**

```
REM Get all available interfaces of the actual device
Dim DeviceInterfaceList As System.Collections.Arraylist = My-
App.myIsl.GetAvailableInterfacesOfDevice()

REM Add the available interfaces to a combobox named CMB_Interfaces
For Each tmpInterface As System.Collections.Generic.SortedDictionary(Of
String, String) in
DeviceInterfaceList
            @ CMB_Interfaces @.Items.Add(tmpInterface('NAME').ToString())
Next
```

## 3.3.2 GetAvailableSinamicsVersionsForInterfaces

This command returns the available SINAMICS firmware versions for the transferred interface type.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| CPUInterface | Transfer of the interface type for which the available SINAMICS firmware versions should be returned |

### Example

Table 3- 13 Code example

**Transfer to a control**

```
Source="__call_GetAvailableSinamicsVersionsForInterfaces('DP1')"
```

**Use in the source code**

```
REM collect all available interfaces for a device and show the available
sinamics versions for every
REM interface

REM Get all available interfaces of the actual device
Dim DeviceInterfaceList As System.Collections.Arraylist = My-
App.myIsl.GetAvailableInterfacesOfDevice()
Dim strOutput As String
Dim SinamicsVersionsList As System.Collections.Arraylist
REM Loop through all available interfaces of the device
For Each tmpInterface As System.Collections.Generic.SortedDictionary(Of
String, String) in DeviceInterfaceList
         REM Add the name of the actual interface to the output string
         strOutput += tmpInterface('NAME') + ':'+ VBCRLF


         REM Get all available Sinamics versions for the actual interface

         SinamicsVersionsList = My-
App.myIsl.GetAvailableSinamicsVersionsForInterfaces(tmpInterface('NAME'))
         REM Loop through all available Sinamics versions
         For Each tmpVersion As Sys-
tem.Collections.Generic.SortedDictionary(Of String, String) in
         SinamicsVersionsList
         REM Add the version to the output string
         strOutput += tmpVersion ('NAME') + VBCRLF
```

**Use in the source code**

```
        Next
        REM Add a separator between interfaces
        strOutput += '######################' + VBCRLF
Next
MsgBox(strOutput)
```

For a SIMOTION D445, the following is output in the message box.

```
DP1:
V2_6_2
V4_4_0
V4_4_5
######################
DP2/MPI:
V2_6_2
V4_4_0
V4_4_5
######################
DP_INT:
Vx_x_x
######################
PN1:
V4_4_0
V4_4_5
######################
```

### 3.3.3 GetAvailableSinamicsExports

This command returns all the available SINAMICS devices in the path **SINAMICS\Devices**. This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

#### Command parameters

This command does not have any parameters.

#### Example

Table 3- 14 Code example

**Transfer to a control**

```
Source="__call_GetAvailableSinamicsExports"
```

**Use in the source code**

```
REM Get all available sinamics exports
Dim SinExportsList As System.Collections.Arraylist = My-
App.myIsl.GetAvailableSinamicsExports()
REM Add the names of the available sinamics exports to a listbox named
LB_SinExports
For Each tmpSinExport As System.Collections.Generic.SortedDictionary(Of
String, String) in SinExportsList
          @LB_ SinExports@.Items.Add(tmpSinExport ('NAME').ToString())
Next
```

## 3.3.4 GetAddressPossibilitiesForInterface

The various options for address assignment of the *ImportSinamicsStation* command are returned with this command, according to the selected interface.
This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key. For an IP address, only the last three positions are returned

### Command parameters

**Transfer parameters**

CPUInterface Transfer of the interface type for which a possible assignment of the address (PROFIBUS or IP address) should be returned.

### Example

Table 3- 15 Code example

**Transfer to a control**

```
Source="__call_GetAddressPossibilitiesForInterface(__ref_CB_Com_Interfaces
.Text)"
```

**Use in the source code**

```
REM Get all possibilities for the last three digits of the IP Address
Dim PNAvailableAddresses As System.Collections.Arraylist = MyApp.myIsl.
GetAddressPossibilitiesForInterface('PN1')
REM Save the last three digits from the user in a string
Dim strUserSelection = @TB_IPAddress4@.Text
REM Flag to signalize that the ip address is already assigned
Dim boAddressAlreadyassigned As As Boolean  = True

REM Loop through all possibilities
For Each tmpPossibleAddress As Sys-
tem.Collections.Generic.SortedDictionary(Of String, String) in
PNAvailableAddresses
        REM Check if the user input is a member of the possiblities list
        If tmpPossibleAddress.ContainsValue(strUserSelection) Then
                REM User selection found in the list so the Address can
be used
                boAddressAlreadyassigned = False
                Exit For
        End IF
Next

If boAddressAlreadyassigned Then
        REM boAddressAlreadyassigned is True means the address is already
assigned
        MsgBox('The address you have selected is already assigned!')
Else
```

**Use in the source code**

```
         REM Import Sinamics Station
         MyApp.NextCommand(444)
End If
```

## 3.3.5 GetNotConnectedDOs

This command returns the names of all drive objects (DO) of a SINAMICS unit that are not connected.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| DeviceName | Transfer of the name of the SINAMICS device whose disconnected DOs should be returned |

### Example

Table 3- 16 Code example

**Transfer to a control**

```
Source="__call_GetNotConnectedDOs(__ref_Cb_SinamcisStationsOfDevice.Text)"
```

**Use in the source code**

```
Dim FreeDOsList As System.Collections.Arraylist = My-
App.myIsl.GetNotConnectedDOs('SINAMICS_INTEGRATED')
```

## 3.3.6 GetNotConnectedTOs

This command returns all technology objects (TO) that are not connected. If a TO has an I/O address above 65000, it will be assumed to be disconnected.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the "EXTERNALTYPENAME" and "NAME" keys.

### Command parameters

This command does not have any parameters.

### Example

Table 3- 17 Code example

**Transfer to a control**

```
Source="__call_GetNotConnectedTOs"
```

**Use in the source code**

```
REM Get all not connected TOs
Dim FreeTOsList As System.Collections.Arraylist = My-
App.myIsl.GetNotConnectedTOs()
```

## 3.3.7 GetSinamicsStationsOfDevice

This command returns all the names of SINAMICS devices subordinate to the active SIMOTION device.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the following keys:

ADDRESS, CPUINTERFACVE, EXTERNALTYPENAME, FILEPATH, IP_ADDRESS, NAME, and VERSION. The keys correspond to the parameter from the ImportSinamicsStation (Page 147) command.

### Command parameters

This command does not have any parameters.

### Example

Table 3- 18 Code example

**Transfer to a control**

```
Source="__call_GetSinamcisStationsOfDevice"
```

**Use in the source code**

```
REM Get all Sinamics stations
Dim SinamicsStationsList As System.Collections.Arraylist = My-
App.myIsl.GetSinamcisStationsOfDevice()
```

## 3.3.8 GetTODOConnections

This command returns all connections between technology objects (TO) and drive objects (DO).

If a TO has an I/O address above 65000, it will be assumed to be disconnected.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the following keys:

DEVICE, DO, and TO. The keys correspond to the parameter from the SetTODOConnection (Page 149) command.

### Command parameters

This command does not have any parameters.

### Example

Table 3- 19 Code example

**Transfer to a control**

```
<!-- Add all not connected DOs to a combobox -->
<Control Action="add"
         Size="100"
         Alignment="HorizontalAlignment.Left"
         items="@Source.DO">
    </Control>
<!-- Add all not connected TOs to a listbox -->
<Control Action="add"
         Size="100"
         Alignment="HorizontalAlignment.Left"
         items="@Source.TO">
</Control>
<!-- Add all informations to a listview -->
   Type="ListView"
   Name="LV_TODOConnections"
   View="Details"
   FullRowSelect="True"
   Location="190,240"
   Size="700, 370"
   Source="__call_GetTODOConnections">

    <ListViewItem Name="DO"
                  Items="@Source.DO"
                  size="200"
                  alignment="HorizontalAlignment.Left">
                  <ListViewSubItem Name ="TO"
                                   Items="@Source.TO"
                                   size="200"
                                   alignment="HorizontalAlignment.Left"
                   />
```

**Transfer to a control**

```
                    <ListViewSubItem Name="Device"
                                     Items="@Source.Device"
                                     size="200"
                                    alignment="HorizontalAlignment.Left"
                     />
     </ListViewItem>
     <Events/>
</Control>
```

**Use in the source code**

```
REM Find the TO connected to the DO selected in a combobox and write the
name of the TO in a textbox
REM call a command that uses the information in the textbox to do some-
thing with the TO

REM Get all Connections between Dos and TOs
Dim DOTOConnectionsList As System.Collections.Arraylist = My-
App.myIsl.GetTODOConnections()

REM Get the DO name
Dim strDOName As String = @CB_DOs@.SelectedItem.ToString()

REM Flag to signalize that the DO is found in the connections list
Dim boDOFound As Boolean  = False


For Each tmpConnection As System.Collections.Generic.SortedDictionary(Of
String, String) in DOTOConnectionsList

         REM compare the selected DO name with the name in the list. Con-
vert the strings to lower case
         REM to catch writing mistakes

         If tmpConnection('DO').ToString().ToLower = strDOName.ToLower
Then
                  @TB_TOName@.Text = tmpConnection('TO').ToString()
                  boDOFound = True
                  Exit For
         End If
Next

If boDOFound Then
         REM DO found. Call the command that does something with the TO
e.g. SetTOConfigData
         MyApp.NextCommand(777)
Else
         REM DO not found.
         MsgBox('The selected DO does not exist or is not connected to a
TO')
End If
```

## 3.3.9 GetDOPowerUnitOrderNo

With this command, the parameter *PowerUnitOrderNo* from the command *ImportDOToDevice* is returned. If there is no *ImportDOToDevice* found with the transferred DO name, the *Default* value is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

**Note**

This command returns the first SINAMICS DO with the found name; i.e. the command only works properly if all DO names in the project are unique.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Name | Name of the DO. |
| Default | Return value, if not found. |

### Example

Table 3- 20 Code example

**Transfer to a control**

```
Text = "__Call_GetDOPowerUnitOrderNo('DO_Discharge2','6SL3120-1TE21-
0Axx')"
```

**Use in the source code**

```
Dim strPowerUnitOrderNr As String = MyApp.MyIsl.GetDOPowerUnitOrderNo('DO_
Discharge2','6SL3120-1TE21-0Axx')
```

## 3.3.10 GetDOParameterValue

This command reads out the currently valid value of a SINAMICS DO and returns it. If a value is not found either in the project or in the database, the *DefaultValue* is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

**Note**

This command returns the first SINAMICS DO with the found name; i.e. the command only works properly if all DO names in the project are unique.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Name | Name of the DO. |
| Parameter | Name of the parameter. |
| DefaultValue | Return value, if not found. |

### Example

Table 3- 21 Code example

**Transfer to a control**

```
Checked="__Call_GetDOParameterValue(DO_Axis_red,P1821[0],False)"
```

**Use in the source code**

```
Dim dblNRegKp As Double = MyApp.MyIsl.GetDOParameterValue('DO_ AxisRed',p1461[0]',1)
```

## 3.3.11 GetNotConnectedDMMDOs

With this command, the names of all drive objects (DO) of a SINAMICS device are returned that expect a Double Motor Module and are not yet connected.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| DeviceName | Name of the SINAMICS device. |

### Example

Table 3- 22 Code example

**Transfer to a control**

```
Source="__call_GetNotConnectedDMMDOs(__ref_Cb_sinamicsStationsOfDevice.Text)"
```

**Use in the source code**

```
REM Before generating a project Check if ther are some Dos that need to be connected to a
DMM
Dim UnconnectedDMMDOsList As System.Collections.Arraylist
Dim strOutputTxt As String = 'Before generating the project the following Dos should be
connected to a DMM' + VbCrLf
UnconnectedDMMDOsList = MyApp.myIsl.GetNotConnectedDMMDOs('Sinamics_Integrated')
If UnconnectedDMMDOsList.Count > 0 Then
         For i As Integer = 0 To UnconnectedDMMDOsList.Count -1
                   strOutputTxt += UnconnectedDMMDOsList(i)( 'NAME') + VbCrLf
         Next
         MsgBox(strOutputTxt)
Else
         REM Generate Project
End If
```

## 3.3.12 GetDOsWithFreeDMMSlot

With this command, the names of all drive objects (DO) of a SINAMICS device are returned that have a free X2 slot.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| DeviceName | Name of the SINAMICS device. |

### Example

Table 3- 23 Code example

**Transfer to a control**

```
Source="__call_GetDOsWithFreeDMMSlot(__ref_Cb_sinamicsStationsOfDevice.Text)"
```

**Use in the source code**

```
REM Ask the user if the project schould be generated even if there are some DOs with a
free DMM Slot
Dim DOsWithFreeSlotList As System.Collections.Arraylist
DOsWithFreeSlotList = MyApp.myIsl.GetDOsWithFreeDMMSlot('Sinamics_Integrated')

Dim strOutputTxt As String = 'Some DOs have free DMM Slots' + VbCrLf + 'Do you want to
continue?'

If DOsWithFreeSlotList.Count > 0 Then
          If MessageBox.Show (strOutputTxt, 'Free DMM Slots',
          MessageBoxButtons.YesNo,MessageBoxIcon.Question) = DialogResult.Yes Then
                    REM Generate Project
          End If
End If
```

## 3.3.13 GetDOsWithConnectedDMMSlot

With this command, the names of all drive objects (DO) (and those connected) of a SINAMICS device are returned whose X2 slot is connected with a drive object.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the "NAME" and "REF_DO_NAME" keys.

### Command parameters

| Parameter | |
|---|---|
| DeviceName | Name of the SINAMICS device. |

### Example

Table 3- 24 Code example

**Transfer to a control**

```
Source="__call_GetDOsWithConnectedDMMSlot(__ref_Cb_sinamicsStationsOfDevice.Text)"
```

**Use in the source code**

```
REM Get all DOs with Connected DMM Slot
Dim DOsWithConnectedSlotList As System.Collections.Arraylist
DOsWithConnectedSlotList  = MyApp.myIsl.GetDOsWithConnectedDMMSlot('Sinamics_Integrated')

REM Objects to represent a ListViewItem and its SubItems
Dim DOItem As ListViewItem
Dim RefDOItem As ListViewItem.ListViewSubItem

REM Start updating the ListView
@LV_DMM_DOs@.BeginUpdate()

REM Add all Dos and Ref_DOs to the listView named LV_DMM_DOs
For Each tmpDO As System.Collections.Generic.SortedDictionary(Of String, String) in
DOsWithConnectedSlotList

          DOItem = @LV_DMM_DOs@.Items.Add(tmpDO ('NAME').ToString())
          RefDOItem = DOItem.SubItems. Add(tmpDO ('REF_DO_NAME').ToString())
Next

REM End updating the ListView
@LV_DMM_DOs@.EndUpdate()
```

The ListView is defined as follows:

```
<Control  Action="add"
          Type="ListView"
          Name=" LV_DMM_DOs "
          View="Details"
          MultiSelect="False"
          FullRowSelect="True"
          HideSelection="True"
          Location="190,240"
          Size="600, 370"
          GridLines ="True">

      <ListViewItem Name="DO"
                    size="300"
                    alignment="HorizontalAlignment.Left" />
             <ListViewSubItem Name="REF_DO"
                              size="300"
                              alignment="HorizontalAlignment.Left" />

       </ListViewItem>
       <Events>
            <SelectedIndexChanged code="" />
        </Events>
</Control>
```

## 3.3.14 GetDOs

This command returns the names of the drive objects (DO) of a SINAMICS device.

This command can be transferred directly to a control property via the *__Call_* prefix. Calling it in the source code returns a collection of *SortedDictionary* objects with the unique "NAME" key.

### Command parameters

**Transfer parameters**

| | |
|---|---|
| Name | Name of the SINAMICS device. |

### Example

Table 3- 25 Code example

**Transfer to a control**

```
Source="__call_ GetDOs(__ref_Cb_sinamicsStationsOfDevice.Text)"
```

**Use in the source code**

```
Dim DOsList As System.Collections.Arraylist
DOsList  = MyApp.myIsl.GetDOs('Sinamics_Integrated')
```

## 3.3.15 GetDOExportName

With this command, the parameter *ExportName* from the command *ImportDOToDevice* is returned. If there is no *ImportDOToDevice* found with the transferred DO name, the *Default* value is returned.

This command can be transferred directly to a control property via the *__Call_* prefix.

**Note**

This command returns the first SINAMICS DO with the found name; i.e. the command only works properly if all DO names in the project are unique.

### Command parameters

| Parameter | |
|---|---|
| Name | Name of the DO. |
| Default | Return value, if not found. |

### Example

Table 3- 26 Code example

**Transfer to a control**

```
DynamicValueSelection="__Call_GetDOExportName('DO_Discharge1','1FK7024-xAK7x-xAxx')"
```

**Use in the source code**

```
Dim DOExportName As String = MyApp.myIsl.GetDOExportName('DO_Discharge1','1FK7024-xAK7x-
xAxx')
```

# Contact

A

## A.1 Contacts

Siemens AG
Digital Factory
Factory Automation
Production Machines
DF FA PMA APC
Frauenauracher Strasse 80
D-91056 Erlangen, Germany
Fax: +49 9131 98 1297
tech.team.motioncontrol@siemens.com

## A.2 Internet addresses

Additional information on various topics is provided on the following Internet pages.

### See also

SIMOTION (www.siemens.com/simotion)

SINAMICS (www.siemens.com/sinamics)

Motion Control / Application Center (www.siemens.com/motioncontrol/apc)

Packaging (www.siemens.com/packaging)