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.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DANGER</strong></td>
<td>indicates that death or severe personal injury <strong>will</strong> result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>WARNING</strong></td>
<td>indicates that death or severe personal injury <strong>may</strong> result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>CAUTION</strong></td>
<td>indicates that minor personal injury can result if proper precautions are not taken.</td>
</tr>
<tr>
<td><strong>NOTICE</strong></td>
<td>indicates that property damage can result if proper precautions are not taken.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WARNING</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

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.
# Table of contents

1  VBS for Creating Procedures and Actions ...................................................................................... 11
   1.1  VBS for creating procedures and actions ................................................................................ 11
   1.2  Using Visual Basic Script in WinCC ....................................................................................... 12
   1.3  Modules and Procedures ........................................................................................................ 15
   1.4  Actions ................................................................................................................................... 18
   1.5  Multiple Use of Procedures and Actions .................................................................................. 20
   1.6  Use of CrossReference ........................................................................................................... 22
   1.7  Using Global Tags in VBS ....................................................................................................... 24
   1.8  VBScript Editors ..................................................................................................................... 26
       1.8.1  VBScript Editors .................................................................................................................. 26
       1.8.2  Global Script Editor ........................................................................................................... 27
       1.8.3  Working in an Editing Window ......................................................................................... 30
       1.8.4  Working with the Toolbars ............................................................................................... 33
       1.8.5  Deleting Actions or Procedures ......................................................................................... 35
   1.9  Creating and Editing Procedures ............................................................................................ 36
       1.9.1  Creating and Editing Procedures ....................................................................................... 36
       1.9.2  Creating a New Procedure ................................................................................................. 39
       1.9.3  How to Write Procedure Codes ....................................................................................... 41
       1.9.4  Using Standard and Project Procedures .......................................................................... 44
       1.9.5  How to add module-related information ........................................................................... 45
       1.9.6  Protecting a Module with a Password .............................................................................. 47
       1.9.7  Saving a Procedure .......................................................................................................... 48
       1.9.8  Renaming a Procedure or Module .................................................................................... 50
   1.10 Creating and Editing Actions .................................................................................................. 52
       1.10.1 Creating and Editing Actions ............................................................................................ 52
       1.10.2 Creating a New Action ....................................................................................................... 56
       1.10.3 How to Edit Actions .......................................................................................................... 57
       1.10.4 How to add action-related information ............................................................................. 60
       1.10.5 Protecting an Action with a Password ............................................................................. 62
       1.10.6 Saving Actions .................................................................................................................. 63
       1.10.7 Triggers ............................................................................................................................. 64
           1.10.7.1 Triggers ......................................................................................................................... 64
           1.10.7.2 Animation trigger ......................................................................................................... 68
           1.10.7.3 How to add a trigger of the type "Timer" ....................................................................... 70
           1.10.7.4 How to add a trigger of the type "Tag" ......................................................................... 72
           1.10.7.5 How to change a trigger .............................................................................................. 74
           1.10.7.6 How to delete a trigger ................................................................................................ 75
           1.10.8 How to Rename an Action ............................................................................................. 76
   1.11 How to activate global actions in Runtime ............................................................................... 78
   1.12 Diagnostics ........................................................................................................................... 80
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.12.1</td>
<td>Diagnostics</td>
<td>80</td>
</tr>
<tr>
<td>1.12.2</td>
<td>GSC Diagnostics</td>
<td>81</td>
</tr>
<tr>
<td>1.12.2.1</td>
<td>GSC Diagnostics</td>
<td>81</td>
</tr>
<tr>
<td>1.12.2.2</td>
<td>Inserting the GSC Diagnostics Window into a Picture</td>
<td>82</td>
</tr>
<tr>
<td>1.12.2.3</td>
<td>GSC Diagnostics Attributes</td>
<td>82</td>
</tr>
<tr>
<td>1.12.2.4</td>
<td>GSC Diagnostics Toolbar</td>
<td>83</td>
</tr>
<tr>
<td>1.12.3</td>
<td>GSC Runtime</td>
<td>84</td>
</tr>
<tr>
<td>1.12.3.1</td>
<td>GSC Runtime</td>
<td>84</td>
</tr>
<tr>
<td>1.12.3.2</td>
<td>How to insert the GSC Runtime Window into a Picture</td>
<td>85</td>
</tr>
<tr>
<td>1.12.3.3</td>
<td>GSC Runtime Attributes</td>
<td>86</td>
</tr>
<tr>
<td>1.12.4</td>
<td>Testing with the Debugger</td>
<td>87</td>
</tr>
<tr>
<td>1.12.4.1</td>
<td>Testing with the Debugger</td>
<td>87</td>
</tr>
<tr>
<td>1.12.4.2</td>
<td>How to Activate the Debugger</td>
<td>89</td>
</tr>
<tr>
<td>1.12.4.3</td>
<td>Principles of Debugging</td>
<td>90</td>
</tr>
<tr>
<td>1.12.4.4</td>
<td>Structure of VBScript Files</td>
<td>92</td>
</tr>
<tr>
<td>1.12.4.5</td>
<td>Action and Procedure Names in the Debugger</td>
<td>94</td>
</tr>
<tr>
<td>1.12.4.6</td>
<td>Selecting a Script for Editing</td>
<td>96</td>
</tr>
<tr>
<td>1.12.4.7</td>
<td>Processing Scripts Step-by-Step</td>
<td>97</td>
</tr>
<tr>
<td>1.12.4.8</td>
<td>Setting Breakpoints</td>
<td>98</td>
</tr>
<tr>
<td>1.12.4.9</td>
<td>Deleting Breakpoints</td>
<td>99</td>
</tr>
<tr>
<td>1.12.4.10</td>
<td>How to Set Bookmarks in Scripts</td>
<td>100</td>
</tr>
<tr>
<td>1.12.4.11</td>
<td>How to Determine and Modify Tag and Property Values</td>
<td>101</td>
</tr>
<tr>
<td>1.12.4.12</td>
<td>Executing Script Commands</td>
<td>102</td>
</tr>
<tr>
<td>1.13</td>
<td>Printing VBScripts</td>
<td>104</td>
</tr>
<tr>
<td>1.14</td>
<td>VBS Reference</td>
<td>105</td>
</tr>
<tr>
<td>1.14.1</td>
<td>VBS Reference</td>
<td>105</td>
</tr>
<tr>
<td>1.14.2</td>
<td>Objects and Lists</td>
<td>107</td>
</tr>
<tr>
<td>1.14.2.1</td>
<td>Objects and Lists</td>
<td>107</td>
</tr>
<tr>
<td>1.14.2.2</td>
<td>Alarm object</td>
<td>110</td>
</tr>
<tr>
<td>1.14.2.3</td>
<td>Alarms object (list)</td>
<td>110</td>
</tr>
<tr>
<td>1.14.2.4</td>
<td>AlarmLogs Object</td>
<td>112</td>
</tr>
<tr>
<td>1.14.2.5</td>
<td>DataItem Object</td>
<td>113</td>
</tr>
<tr>
<td>1.14.2.6</td>
<td>DataLogs Object</td>
<td>114</td>
</tr>
<tr>
<td>1.14.2.7</td>
<td>DataSet Object (List)</td>
<td>116</td>
</tr>
<tr>
<td>1.14.2.8</td>
<td>HMI RunTime Object</td>
<td>118</td>
</tr>
<tr>
<td>1.14.2.9</td>
<td>Item Object</td>
<td>119</td>
</tr>
<tr>
<td>1.14.2.10</td>
<td>Layer Object</td>
<td>120</td>
</tr>
<tr>
<td>1.14.2.11</td>
<td>Layers Object (Listing)</td>
<td>121</td>
</tr>
<tr>
<td>1.14.2.12</td>
<td>Logging Object</td>
<td>122</td>
</tr>
<tr>
<td>1.14.2.13</td>
<td>ProcessValue Object</td>
<td>123</td>
</tr>
<tr>
<td>1.14.2.14</td>
<td>ProcessValues Object (List)</td>
<td>124</td>
</tr>
<tr>
<td>1.14.2.15</td>
<td>Project Object</td>
<td>124</td>
</tr>
<tr>
<td>1.14.2.16</td>
<td>ScreenItem Object</td>
<td>125</td>
</tr>
<tr>
<td>1.14.2.17</td>
<td>ScreenItems Object (List)</td>
<td>129</td>
</tr>
<tr>
<td>1.14.2.18</td>
<td>Screen Object</td>
<td>131</td>
</tr>
<tr>
<td>1.14.2.19</td>
<td>Screens Object (List)</td>
<td>134</td>
</tr>
<tr>
<td>1.14.2.20</td>
<td>SmartTags Object</td>
<td>136</td>
</tr>
<tr>
<td>1.14.2.21</td>
<td>Tag Object</td>
<td>137</td>
</tr>
<tr>
<td>1.14.2.22</td>
<td>Tags Object (List)</td>
<td>140</td>
</tr>
<tr>
<td>1.14.2.23</td>
<td>TagSet Object (List)</td>
<td>141</td>
</tr>
<tr>
<td>1.14.3</td>
<td>Object types of the ScreenItem object</td>
<td>143</td>
</tr>
</tbody>
</table>
**WinCC: Scripting (VBS, ANSI-C, VBA)**

1.14 Examples

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.14.3.1 Object types of the ScreenItem object</td>
<td>143</td>
</tr>
<tr>
<td>1.14.3.2 Standard objects</td>
<td>144</td>
</tr>
<tr>
<td>1.14.3.3 Smart objects</td>
<td>169</td>
</tr>
<tr>
<td>1.14.3.4 Windows objects</td>
<td>201</td>
</tr>
<tr>
<td>1.14.3.5 Tube objects</td>
<td>215</td>
</tr>
<tr>
<td>1.14.3.6 Controls</td>
<td>218</td>
</tr>
<tr>
<td>1.14.3.7 Customized Object</td>
<td>298</td>
</tr>
<tr>
<td>1.14.3.8 Group</td>
<td>300</td>
</tr>
<tr>
<td>1.14 Properties</td>
<td>301</td>
</tr>
<tr>
<td>1.14.4 Properties</td>
<td>301</td>
</tr>
<tr>
<td>1.14.4.2 A</td>
<td>302</td>
</tr>
<tr>
<td>1.14.4.3 B</td>
<td>321</td>
</tr>
<tr>
<td>1.14.4.4 C</td>
<td>362</td>
</tr>
<tr>
<td>1.14.4.5 D</td>
<td>397</td>
</tr>
<tr>
<td>1.14.4.6 E</td>
<td>407</td>
</tr>
<tr>
<td>1.14.4.7 F</td>
<td>416</td>
</tr>
<tr>
<td>1.14.4.8 G</td>
<td>438</td>
</tr>
<tr>
<td>1.14.4.9 H</td>
<td>442</td>
</tr>
<tr>
<td>1.14.4.10 I</td>
<td>451</td>
</tr>
<tr>
<td>1.14.4.11 L</td>
<td>457</td>
</tr>
<tr>
<td>1.14.4.12 M</td>
<td>488</td>
</tr>
<tr>
<td>1.14.4.13 N</td>
<td>509</td>
</tr>
<tr>
<td>1.14.4.14 O</td>
<td>512</td>
</tr>
<tr>
<td>1.14.4.15 P</td>
<td>528</td>
</tr>
<tr>
<td>1.14.4.16 Q</td>
<td>546</td>
</tr>
<tr>
<td>1.14.4.17 R</td>
<td>547</td>
</tr>
<tr>
<td>1.14.4.18 S</td>
<td>557</td>
</tr>
<tr>
<td>1.14.4.19 T</td>
<td>595</td>
</tr>
<tr>
<td>1.14.4.20 U</td>
<td>671</td>
</tr>
<tr>
<td>1.14.4.21 V</td>
<td>680</td>
</tr>
<tr>
<td>1.14.4.22 W</td>
<td>697</td>
</tr>
<tr>
<td>1.14.4.23 X - Z</td>
<td>702</td>
</tr>
<tr>
<td>1.14.5 Methods</td>
<td>710</td>
</tr>
<tr>
<td>1.14.5.1 Methods</td>
<td>710</td>
</tr>
<tr>
<td>1.14.5.2 Methods A to E</td>
<td>712</td>
</tr>
<tr>
<td>1.14.5.3 Get methods</td>
<td>720</td>
</tr>
<tr>
<td>1.14.5.4 Methods H to M</td>
<td>769</td>
</tr>
<tr>
<td>1.14.5.5 Methods N to R</td>
<td>779</td>
</tr>
<tr>
<td>1.14.5.6 Methods S to T</td>
<td>797</td>
</tr>
<tr>
<td>1.14.5.7 Methods U to Z</td>
<td>812</td>
</tr>
<tr>
<td>1.14.6 Appendix</td>
<td>820</td>
</tr>
<tr>
<td>1.14.6.1 Error Messages from Database Area</td>
<td>820</td>
</tr>
<tr>
<td>1.15 Examples of VBScript</td>
<td>823</td>
</tr>
<tr>
<td>1.15.1 Examples of VBScript</td>
<td>823</td>
</tr>
<tr>
<td>1.15.2 Examples in WinCC</td>
<td>823</td>
</tr>
<tr>
<td>1.15.2.1 Examples in WinCC</td>
<td>823</td>
</tr>
<tr>
<td>1.15.2.2 Example: Accessing objects in Graphics Designer</td>
<td>824</td>
</tr>
<tr>
<td>1.15.2.3 Example: Defining the color of objects</td>
<td>825</td>
</tr>
<tr>
<td>1.15.2.4 Example: How to Configure Language Changes</td>
<td>825</td>
</tr>
<tr>
<td>1.15.2.5 Example: Deactivating Runtime</td>
<td>826</td>
</tr>
<tr>
<td>1.15.2.6 Example: Configuring change picture globally</td>
<td>826</td>
</tr>
<tr>
<td>1.15.2.7 Example: Configuring Change Picture Via Property</td>
<td>827</td>
</tr>
</tbody>
</table>
Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.15.2.8 Example: Configuring diagnostics output via Trace</td>
<td>827</td>
</tr>
<tr>
<td>1.15.2.9 Example: Writing tag values</td>
<td>828</td>
</tr>
<tr>
<td>1.15.2.10 Example: How to Read Tag Values</td>
<td>830</td>
</tr>
<tr>
<td>1.15.2.11 Example: Writing Object Properties</td>
<td>833</td>
</tr>
<tr>
<td>1.15.2.12 Example: How to Start an Action on the Server (Logging Object)</td>
<td>836</td>
</tr>
<tr>
<td>1.15.2.13 Example: You connect or disconnect a database backup with VBS.</td>
<td>838</td>
</tr>
<tr>
<td>1.15.2.14 Dynamization of Controls</td>
<td>841</td>
</tr>
<tr>
<td>1.15.3 General Examples</td>
<td>855</td>
</tr>
<tr>
<td>1.15.3.1 General examples for VBScript</td>
<td>855</td>
</tr>
<tr>
<td>1.15.3.2 Example: Configuring a Database Connection with VBS</td>
<td>856</td>
</tr>
<tr>
<td>1.15.3.3 Example: Using the MS Automation interface</td>
<td>858</td>
</tr>
<tr>
<td>1.15.3.4 Example: Starting an external application</td>
<td>859</td>
</tr>
<tr>
<td>1.16 Basic Principles of VBScript</td>
<td>861</td>
</tr>
<tr>
<td>1.16.1 Basic Principles of VBScript</td>
<td>861</td>
</tr>
<tr>
<td>1.16.2 VBScript Basics</td>
<td>861</td>
</tr>
<tr>
<td>2 ANSI-C for Creating Functions and Actions</td>
<td>863</td>
</tr>
<tr>
<td>2.1 Creating Functions and Actions with ANSI-C</td>
<td>863</td>
</tr>
<tr>
<td>2.2 Creating Functions and Actions</td>
<td>864</td>
</tr>
<tr>
<td>2.3 Characteristics of Project Functions</td>
<td>867</td>
</tr>
<tr>
<td>2.4 Characteristics of Standard Functions</td>
<td>868</td>
</tr>
<tr>
<td>2.5 Characteristics of Internal Functions</td>
<td>870</td>
</tr>
<tr>
<td>2.6 Characteristics of Local Actions</td>
<td>871</td>
</tr>
<tr>
<td>2.7 Characteristics of Global Actions</td>
<td>872</td>
</tr>
<tr>
<td>2.8 How to Add Global Script Runtime to a Project's Startup List</td>
<td>873</td>
</tr>
<tr>
<td>2.9 Use of Global C-Tags</td>
<td>874</td>
</tr>
<tr>
<td>2.10 Use of DLLs in Functions and Actions</td>
<td>876</td>
</tr>
<tr>
<td>2.11 The Global Script Editor</td>
<td>878</td>
</tr>
<tr>
<td>2.11.1 The Global Script Editor</td>
<td>878</td>
</tr>
<tr>
<td>2.11.2 Working in the Edit Window</td>
<td>880</td>
</tr>
<tr>
<td>2.11.2.1 Working in the Edit Window</td>
<td>880</td>
</tr>
<tr>
<td>2.11.2.2 Editing Functions with the Keyboard</td>
<td>881</td>
</tr>
<tr>
<td>2.11.2.3 Editing Functions with the Mouse</td>
<td>882</td>
</tr>
<tr>
<td>2.11.3 Working with the Toolbars</td>
<td>882</td>
</tr>
<tr>
<td>2.11.4 How to Set Different Views</td>
<td>885</td>
</tr>
<tr>
<td>2.11.5 How to Set the Font Style</td>
<td>885</td>
</tr>
<tr>
<td>2.11.6 How to Use &quot;Save As...&quot;</td>
<td>886</td>
</tr>
<tr>
<td>2.11.7 How to Delete Actions or Project and Standard Functions</td>
<td>887</td>
</tr>
<tr>
<td>2.11.8 How to Generate a New Header</td>
<td>887</td>
</tr>
<tr>
<td>2.11.9 How to Compile All Functions</td>
<td>888</td>
</tr>
<tr>
<td>2.11.10 How to Search in Files</td>
<td>889</td>
</tr>
<tr>
<td>2.11.11 Printing Functions and Actions</td>
<td>890</td>
</tr>
<tr>
<td>2.11.11.1 Printing Functions and Actions</td>
<td>890</td>
</tr>
<tr>
<td>2.11.11.2 How to Set the Print Parameters</td>
<td>890</td>
</tr>
<tr>
<td>2.11.11.3 How to Open Page View</td>
<td>891</td>
</tr>
<tr>
<td>2.11.11.4 How to Print the Project Documentation</td>
<td>891</td>
</tr>
</tbody>
</table>
### Table of contents

2.12 Creating and Editing Functions .......................................................... 892
  2.12.1 Creating and Editing Functions .................................................. 892
  2.12.2 How to Create a New Function ................................................. 895
  2.12.3 How to Write Function Code .................................................. 896
  2.12.4 How to Use Internal Functions .............................................. 897
  2.12.5 How to Use Standard and Project Functions ......................... 897
  2.12.6 Inserting Additional Function-Related Information ............... 898
  2.12.7 How to Protect a Function Against Unauthorized Access ....... 900
  2.12.8 How to Compile and Save a Function ...................................... 901
  2.12.9 How to Rename a Function ................................................... 902
  2.12.10 How to Use Functions from Other Sources ........................... 903

2.13 Creating and Editing Actions ......................................................... 905
  2.13.1 How To Create and Edit Actions .............................................. 905
  2.13.2 WinCC Coding Rule .............................................................. 908
  2.13.3 How to Create a New Action .................................................. 909
  2.13.4 How to Edit Actions ............................................................. 910
  2.13.5 How to add action-related information ................................. 910
  2.13.6 How to Protect an Action Against Unauthorized Access ....... 912
  2.13.7 How to Compile and Save an Action ..................................... 913
  2.13.8 Triggers .................................................................................. 914
    2.13.8.1 Triggers ............................................................................ 914
    2.13.8.2 How to Add a New Trigger of the "Timer" Type .............. 917
    2.13.8.3 How to Add a New Trigger of the "Tag" Type ................. 918
    2.13.8.4 How to change a trigger ................................................ 920
    2.13.8.5 How to delete a trigger .................................................. 921
    2.13.9 How to Assign Authorizations .......................................... 922
    2.13.10 How to Export an Action ................................................... 922
    2.13.11 How to Import an Action ................................................... 923
    2.13.12 How to Rename an Action ................................................ 924
    2.13.13 How to Use Actions From Other Sources ......................... 925

2.14 Runtime Behavior of Actions .......................................................... 927
  2.14.1 Runtime Behavior of Actions ................................................... 927
  2.14.2 GSC Runtime .......................................................................... 928
    2.14.2.1 GSC Runtime ................................................................ 928
    2.14.2.2 How to Place GSC Runtime in a Process Picture .......... 931
  2.14.2.3 Attributes of GSC Runtime .............................................. 932
  2.14.2.4 How to Edit Actions ......................................................... 932
  2.14.3 GSC Diagnose ......................................................................... 933
    2.14.3.1 GSC Diagnose ................................................................ 933
    2.14.3.2 How to Place GSC Diagnose in a Process Picture ......... 934
  2.14.3.3 Attributes of GSC Diagnose .............................................. 935
  2.14.3.4 The Toolbar of GSC Diagnose ........................................ 935

2.15 ANSI-C function descriptions ......................................................... 937
  2.15.1 lpszPictureName ................................................................. 937
  2.15.2 Standard functions ............................................................... 937
    2.15.2.1 Standard functions - short description ......................... 937
    2.15.2.2 Alarm ............................................................................ 938
    2.15.2.3 Graphics ......................................................................... 942
    2.15.2.4 Obsolete functions ...................................................... 947
  2.15.2.5 Report ............................................................................... 999
### 3 VBA for Automated Configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Automated configuration</td>
<td>1613</td>
</tr>
<tr>
<td>3.2</td>
<td>Introduction: Using VBA in WinCC</td>
<td>1614</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Introduction: Using VBA in WinCC</td>
<td>1614</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Differentiation: Using VBA</td>
<td>1614</td>
</tr>
</tbody>
</table>
Table of contents

3.2.3 Organizing VBA Code in a WinCC Project ................................................................. 1615
3.2.4 How to export and import VBA code ........................................................................ 1618
3.2.5 Executing VBA Macros in Graphics Designer ....................................................... 1619
3.3 VBA in the Graphics Designer .................................................................................. 1621
  3.3.1 VBA in the Graphics Designer ............................................................................. 1621
  3.3.2 Adapting the Graphics Designer with VBA ............................................................ 1623
  3.3.2.1 Adapting the Graphics Designer with VBA ...................................................... 1623
  3.3.2.2 Language-Dependent Configuration with VBA .............................................. 1624
  3.3.2.3 Creating Customized Menus and Toolbars ...................................................... 1626
  3.3.2.4 Accessing the component library with VBA ..................................................... 1646
  3.3.3 Editing Pictures with VBA ..................................................................................... 1654
  3.3.3.1 Editing Pictures with VBA ............................................................................. 1654
  3.3.3.2 How to Create Picture-specific Menus and Toolbars .................................. 1655
  3.3.3.3 Editing Layers with VBA ................................................................................ 1657
  3.3.3.4 Editing a Copy of a Picture with VBA ............................................................... 1658
  3.3.4 Editing Objects with VBA ..................................................................................... 1660
  3.3.4.1 Editing Objects with VBA ............................................................................. 1660
  3.3.4.2 Default objects, Smart objects, Windows objects and Tube objects ............ 1662
  3.3.4.3 Group Objects ................................................................................................. 1674
  3.3.4.4 Customized Objects ....................................................................................... 1682
  3.3.5 Creating Dynamics with VBA .............................................................................. 1686
  3.3.5.1 Creating Dynamics with VBA ....................................................................... 1686
  3.3.5.2 Configuring Dynamics in the Properties of Pictures and Objects ................ 1687
  3.3.5.3 Configuring Event-Driven Actions with VBA ............................................... 1699
  3.3.5.4 Editing Triggers .............................................................................................. 1707
  3.3.6 Event Handling ...................................................................................................... 1709
  3.3.7 Accessing External Applications with VBA ......................................................... 1713
  3.3.7.1 Accessing External Applications with VBA .................................................... 1713
  3.3.7.2 Example: Accessing MS Excel with VBA ...................................................... 1714
  3.4 VBA in the WinCC Configuration Studio .................................................................. 1725
  3.4.1 VBA in the WinCC Configuration Studio .............................................................. 1725
  3.4.2 Example: VBA in the WinCC Configuration Studio ............................................ 1726
  3.5 AddIns ....................................................................................................................... 1734
  3.5.1 AddIns .................................................................................................................. 1734
  3.5.2 Linking Add Ins ..................................................................................................... 1734
  3.5.3 How to Configure an AddIn in the Graphics Designer ......................................... 1737
  3.5.4 Example: Creating Add Ins ................................................................................ 1738
  3.5.4.1 Example: Creating Add Ins ............................................................................... 1738
  3.5.4.2 Example: Creating an Add In with Visual Basic 6.0 ........................................ 1739
  3.6 VBA Reference ......................................................................................................... 1745
  3.6.1 The object model of the Graphics Designer ......................................................... 1745
  3.6.1.1 VBA Reference ............................................................................................... 1745
  3.6.1.2 VBA Reference: ActionDynamic .................................................................... 1747
  3.6.1.3 VBA Reference: HMIObjets .......................................................................... 1749
  3.6.1.4 VBA Reference: Languages .......................................................................... 1751
  3.6.1.5 Events ............................................................................................................. 1752
  3.6.1.6 Methods .......................................................................................................... 1785
  3.6.1.7 Objects and Lists ............................................................................................ 1894
  3.6.1.8 Properties ....................................................................................................... 2106
  3.6.2 VBA in Other WinCC Editors ............................................................................... 2560
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6.2.1</td>
<td>VBA in Other WinCC Editors</td>
<td>2560</td>
</tr>
<tr>
<td>3.6.2.2</td>
<td>VBA in Tag Management</td>
<td>2562</td>
</tr>
<tr>
<td>3.6.2.3</td>
<td>VBA im Tag Logging</td>
<td>2573</td>
</tr>
<tr>
<td>3.6.2.4</td>
<td>VBA in the Text Library</td>
<td>2608</td>
</tr>
<tr>
<td>3.6.2.5</td>
<td>VBA in Alarm Logging</td>
<td>2622</td>
</tr>
<tr>
<td></td>
<td>Index</td>
<td>2637</td>
</tr>
</tbody>
</table>
1.1 VBS for creating procedures and actions

Contents

WinCC provides the possibility of dynamizing the Runtime environment using the Visual Basic Script. It is possible to use VBS to program global actions and procedures as well as dynamizing graphic objects and triggering actions in Runtime.

This chapter will show you

- How to work with VBScript editors
- How to create and edit procedures
- How to create and edit actions
- How to activate VBScripts in Runtime
- How to execute diagnostics on scripts in Runtime
- The object model of graphic Runtime system
- Detailed examples on using VBScript
1.2 Using Visual Basic Script in WinCC

Introduction

In addition to the C script, WinCC also provides the VBScript program language as a programming interface in order to make the WinCC Runtime environment dynamic.

Target Group of the Documentation

This documentation is aimed at project engineers with experience of Visual Basic or WinCC Scriptings (C) used to date.

Application Options

VBScript (VBS) provides access to tags and objects of the graphical Runtime system at Runtime and can execute picture-independent functions:

- Tags: Tag values can be read and written in order, for example, to specify tag values for the PLC by clicking the mouse when positioned on a button.
- Objects: Object properties can be made dynamic using actions and actions can be triggered by events influencing objects.
- Picture-independent Actions: Picture-independent actions can be triggered cyclically or according to tag values, e.g. for the daily transfer of values into an Excel table.

VBS can be used at the following points in WinCC:

- In the Global Script Editor: This is used to configure picture-independent actions and procedures. The procedures can be used in picture-dependent and picture-independent actions. Several procedures are compiled in a topic-related module.
- In Graphics Designer: Picture-dependent actions can be configured with which the properties of graphic objects can be made dynamic or caused to respond to events in Runtime.
- In user-defined menus and toolbars: Here you configure procedures called in Runtime using the menu and toolbars.

Note

Updating Changed Configuration in Runtime

A changed VB script that is connected with "Menus and toolbars" is only updated after Runtime is restarted.

If you change the properties of "Menus and toolbars" in Runtime, the changes are only applied in the following cases:

- After a picture change, if the configuration change does not affect the basic picture.
- When you load another configuration file and reload the modified configuration file.
Registered Tags in Menus and Toolbars

The registered tags in the scripts of "Menus and toolbars" remain registered when you unselect the picture. If you read indirectly from a process mapping the tags are registered and unregistered again when you unselect the picture. However, registered tags in the scripts of "Menus and toolbars" remain registered when you unselect the picture.

Application Scenarios

VBS can be used in Runtime, for example:

- to configure setpoint value specification for tags for the operation of a graphic object in order to define a value for the PLC by clicking a mouse, for example.
- to configure switching the Runtime language for the operation of a graphic object.
- to configure the change of color, e.g. cyclically (flashing) or to display statuses (motor on).

Apart from the specific WinCC applications, the general functionality of VBS can also be used to customize the Windows environment, e.g.:

- to transfer data to another application (e.g. Excel).
- to start external applications from WinCC.
- to create files and folders.

The automation objects in your environment are available with which to customize the Windows environment.

Note

All the objects supplied with the Windows Script Host (WSH) from Microsoft can be integrated in the environment using the standard VBS method CreateObject. However, there is no direct access to the WSH object itself using VBS from WinCC.

There is no guarantee nor WinCC support for the VBS functionality with regard to its adaptation to the Windows environment.

Limits to Other Programming Languages in WinCC

VBS and C

VBScript can be used in WinCC parallel to C-Script, but do not mix the script types:

- VBScripts and C-scripts can be configured within a picture and project.
- C-scripts cannot be invoked in VBScripts and vice versa.
- VBScript provides internal interfaces to tags and picture objects while the C environment enables access to other WinCC subsystems (e.g. the report system).

VBS and VBA

VBA is used in WinCC Configuration during the configuration in order to adapt Graphics Designer to your individual requirements and to simplify and automate configuration. VBA programs only run in the WinCC configuration environment.
As opposed to VBA, VB scripts only run in WinCC Runtime and, from there, enable access to graphic objects and tags. Objects and pictures can be neither created nor modified on a permanent basis in VBS, as opposed to VBA.

The main language-related differences between VBA and VBS are e.g.:

- VBS was developed for use in Internet, VBA for the automation of software applications.
- The data type of VBS tags is always VARIANT. VBA, on the other hand, differentiates the individual data types such as INT, DOUBLE, STRING, etc.
- Certain language constructs from VBA have been removed from or added to VBS.
- Errors are handled differently in VBS compared to VBA.

A complete list of the differences between VBA and VBS is provided in the Appendix in "Basic Principles of VBScript".

### Procedures, Modules and Actions

VBS in WinCC allows the use of procedures, modules and actions to make the Runtime environment dynamic:

- **Procedures**: Codes are stored in procedures which are then used at several points in the configuration. Retrieve the code or another procedure in an action by invoking the procedure name. Procedures can be created in WinCC with or without return values. Procedures do not have their own trigger, they are always retrieved by an action.

- **Modules**: It is advantageous to compile related procedures to units in modules. Create modules for procedures, for example, which must be used in a specific picture or belong to a specific topic, such as auxiliary mathematical functions or database access functions.

- **Actions**: Actions are always activated by a trigger, namely a triggering event. Actions are configured in graphic object properties, in events which occur on a graphic object or globally in a project. Codes used several times can be called, in the form of procedures, in actions.

---

![Diagram of Procedure, Module, and Action](image)

- **Trigger**
- **Module**
  - Procedure A
  - Procedure B
- **Action**
  - Instruction-1
  - Instruction-2
  - Procedure C
  - Instruction-a
  - Instruction-b
- **Procedure C**
  - Instruction-c
- **Procedure D**
  - Instruction-d
- **Procedure E**

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
1.3 Modules and Procedures

Introduction

Procedures are used to make code, created only once, available at several points in a project. Instead of entering the code several times, simply call in the corresponding procedure. The code is clearer and easier to maintain.

Related procedures should be stored in modules. When a certain procedure is retrieved via an action in Runtime, the module that contains that procedure is loaded. Observe the following when structuring the modules and procedures:

- The more modules that must be loaded when a picture is called in, the worse the performance in Runtime.
- The larger a module, the more procedures are contained and, thus, the longer the loading time for the module.

Organize the modules sensibly, e.g. a module with procedures for a specific system part/picture.

Another type of structuring of procedures in modules is the functional structuring, e.g. a module containing mathematical functions. Use this structure, for example, for modules which should be used globally for projects. The following example illustrates a module containing mathematical functions derived from the standard functions:
Procedure Features

The procedures in WinCC have the following properties:

- They are created and modified by the user.
- They can be protected against modification and viewing by means of a password.
- They do not have a trigger.
- They are stored in a module.

WinCC does not provide predefined procedures, but does provide code templates and Intellisense, for example, to simplify programming. Procedures differ according to module assignment in:

- Standard procedures apply globally to projects located on the computer on which they were created.
- Project procedures can only be used within the project in which they were created.
Module Features

A module is a file in which one or procedures is stored. The modules in WinCC have the following properties:

- They can be protected against modification and viewing by means of a password.
- The have the file extension *.bmo.

Modules differ according to the validity of their procedures in:

- Standard Modules: Contain procedures which are globally available to the project. Standard modules are stored in the WinCC file system under: `<WinCC-Installationsverzeichnis>\ApLib\ScriptLibStd\<Modulname>.bmo`
- Project Modules: Contain project-specific procedures. Project modules are stored in the WinCC file system under: `<Projektverzeichnis>\ScriptLib\<Modulname>.bmo`. Since the project modules are stored in the project directory, they are copied when the WinCC project is copied.

Note

If WinCC must be reinstalled and the standard procedures and modules need to be used again, save the module files prior to the reinstallation in a different directory and copy them in the relevant WinCC directory after the reinstallation has been completed. Otherwise, the standard modules in the WinCC installation directory are deleted during the installation routine.

Using Procedures and Modules

Procedures are used in:

- Actions (in Graphics Designer and Global Script)
- Other procedures (in Global Script)
- User-defined menus and toolbars

Procedures are structured in modules.

See also

Creating and Editing Procedures (Page 36)
VBScript Editors (Page 26)
Basic Principles of VBScript (Page 861)
Actions (Page 18)
Using Visual Basic Script in WinCC (Page 12)
1.4 Actions

Introduction

An action is always started by a trigger. An action, for example, is triggered in Runtime when an object is operated by a mouse click, a certain time has occurred or a tag has been modified.

Action Features

Actions are defined once in Global Script and then available independent of the picture. Global Script actions are only valid in the project in which they were defined. Actions linked to a graphic object are only valid in the picture in which they were defined.

Note

VBS does not currently allow the creation of computer-specific actions.
The following applies for clients in a multi-user system: All global actions configured on a server are also executed on a client when a project is opened.
The following applies to clients in a distributed system: If actions should be used on a client computer, copy all the action files in the corresponding project directory on the client.

Actions have the following properties:

- Actions are created and modified by the user.
- Actions in Global Script can be protected against modification and viewing by means of a password.
- Actions have at least one trigger.
- Actions in Global Script have the file extension *.bac.
- Global Script actions are stored in the WinCC file system under: <Projektverzeichnis>\ScriptAct\Aktionsname.bac

Action Trigger

Triggers are required to execute actions in Runtime. A trigger is linked to an action thus forming the triggering event which calls the action. Actions without triggers will not be carried out.

The following trigger types are available in WinCC:

- Timer: Acyclic or cyclic trigger, e.g. for calling a picture or every hour.
- Tags: Change of value
- Event: Modification of object properties (e.g. change of color) or event on an object (e.g. mouse click).
Processing of Actions in Runtime

**In Graphics Designer**

Two actions of the same type can be executed simultaneously in Runtime. In order, for example, that cyclic actions are not hindered by an action executed by a mouse click, event triggered actions and cyclic/tag triggered actions in Graphics Designer are executed independently of each other.

---

**Note**

Please note that synchronization between both action types in WinCC may only be executed by the DataSet object or by internal WinCC tags. As a result of the separate processing, no common data area exists between event triggered and cyclic/tag triggered actions.

---

If processing cyclic actions in pictures, for example, is prevented by a high system load or another action, the action is started once at the next opportunity. Cycles which are not executed are not retained in a queue but rejected.

After a change of picture, scripts still running are automatically stopped 1 minute after the change of picture.

Scripts which are still running when Runtime is terminated are stopped after 5 seconds.

---

**In Global Script**

Picture-independent actions from Global Script are executed in Runtime in succession after being triggered. If an action is triggered while another action is in progress, the second action is retained in a queue until it can be executed.

---

**Note**

Please note that synchronization between actions in Global Script and in Graphics Designer may only be executed by the DataSet object or by internal WinCC tags. There is no common data area between the actions in Graphics Designer and in Global Script.

---

**Using the Actions**

Actions can be used as follows:

- In Global Script: The global actions defined here run picture-independent in Runtime.
- In Graphics Designer: The actions defined here only run in the configured picture. An action is configured in Graphics Designer on an object property or an event on a graphic object.

---

**See also**

Creating and Editing Actions (Page 52)
Basic Principles of VBScript (Page 861)
Modules and Procedures (Page 15)
Using Visual Basic Script in WinCC (Page 12)
1.5 Multiple Use of Procedures and Actions

Introduction

An action configured with VBS in WinCC is always valid for the project in which it was defined. Procedures have the following areas of application:

- Standard procedures apply globally to projects located on the computer on which they were created.
- Project procedures can only be used in the project in which they were created. If a project is copied, the project procedures (modules) are copied together with the project.

Multiple Use of Procedures and Actions

If actions or procedures/modules are to be used in other projects or on other computers, it is possible either to:

- use the "Save As" function to store the action or module in a different project directory or, for example, on a disk.
- to copy the action or module file in Windows Explorer and paste it in the corresponding project or standard directory on the target computer.

The properties and trigger configured are retained during copying. Copied modules are directly available in Runtime. Copied actions are executed in Runtime after they have been opened and stored once.

Note

Tags used in an action or procedure must also be available on the target computer. If the tag is not available, the action or procedure is not executed.

Procedures which are called in an action must be available on the target computer. If the procedure is not available, a Runtime error occurs during Runtime.

Storing Procedures

If procedures need to be copied in other project directories in order to be able to use them in other projects or different computers, observe the storage path of the procedures in the WinCC file system:

- Standard procedures: <WinCC-Installationsverzeichnis>\ApLib\ScriptLibStd \Modulname.bmo
- Project procedures: <Projektverzeichnis>\ScriptLib\Modulname.bmo

Note

Since procedures are always stored in modules, always copy the module (*.bmo) in which the procedure is contained.
The copied procedures/modules are visible after updating the Global Script navigation window (context menu command "Update") or restarting the editor.

**Storing Actions**

If actions need to be copied in other project directories in order to be able to use then in other projects or different computers, observe the storage path of the actions in the WinCC file system:

```
<Projektverzeichnis>\Script\Aktionsname.bac
```

Each action is stored in a separate file. When an action is copied, all the triggers related to it are also copied.

**Note**

Only actions created in Global Script are stored in the WinCC file system. Actions which are programmed in Graphics Designer are always stored with the current picture and cannot be transferred individually. If a Graphics Designer picture is copied into another project directory, the actions stored with the picture are also copied.

The copied actions are visible after updating the Global Script navigation window (context menu command "Update") or restarting the editor.

**See also**

- Modules and Procedures (Page 15)
- Renaming a Procedure or Module (Page 50)
- Saving a Procedure (Page 48)
- Protecting a Module with a Password (Page 47)
- How to add module-related information (Page 45)
- Using Standard and Project Procedures (Page 44)
- How to Write Procedure Codes (Page 41)
- Creating a New Procedure (Page 39)
- Creating and Editing Procedures (Page 36)
1.6 Use of CrossReference

CrossReference and Tag Trigger

The CrossReference from WinCC can be used to quickly find all the application points of tags, even in VBS actions. Tag triggers in actions in Graphics Designer can be "linked" using CrossReference, i.e. replaced by other tags at all or selected points.

Note

Tags can also be directly linked in Graphics Designer by marking the graphic object and selecting the "Linking ..." command from the shortcut menu.

Further information on CrossReference is available in the WinCC documentation.

Actions and CrossReference

All the actions used in a picture can be displayed by means of the picture properties. To do this mark the picture in WinCC Explorer and select the "Properties" shortcut menu command. After double clicking on an entry, detailed information on the type of dynamics appears.

It is also possible to display all the tags and pictures used in actions by means of the WinCC CrossReference. CrossReference can also be used for the to link tag connections of Graphics Designer actions easily.

Tags and CrossReference

All tags addressed with the following standard formulation are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.

' VBS1
HMIRuntime.Tags("Tagname")

If tags are addressed with different formulations in the code, this can be notified by the following section of the CrossReference:

' WINCC:TAGNAME_SECTION_START
Const TagNameInAction = "TagName"
' WINCC:TAGNAME_SECTION_END

The section can be inserted in VBS actions as often as required.

Note

It is not possible to guarantee the compilation of combined tag names from the CrossReference.
Pictures and CrossReference

All pictures addressed with the following standard formulation are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.

\[ \text{'VBS2}\]
\[ \text{HMIRuntime.BaseScreenName = "Screenname"} \]

If pictures are addressed with different formulations in the code, this can be notified by the following section of the CrossReference:

\[' WINCC:SCREENNAME_SECTION_START\]
\[ Const \text{ ScreenNameInAction = "ScreenName"} \]
\[' WINCC:SCREENNAME_SECTION_END\]

The section can be inserted in VBS actions as often as required.

**Note**

Always enter picture names without the extension "PDL" for reasons of compatibility with future versions.

**See also**

VBS Reference (Page 105)
VBScript Editors (Page 26)
Basic Principles of VBScript (Page 861)
Actions (Page 18)
Modules and Procedures (Page 15)
Using Visual Basic Script in WinCC (Page 12)
1.7 Using Global Tags in VBS

Introduction

Global tags can be defined in the Global Script Editor which can then be used in all actions and procedures.

Using Global Tags in Graphics Designer and Global Script

Observe the following conditions when using global tags in Graphics Designer and Global Script:

- In order to use a global tag in an action in Graphics Designer, call in the procedure in which the tag is defined so that the associated module is loaded in Runtime.

- In order to use a global tag in an action in Global Script, at least one procedure must be activated from the module in at least one global action in which the tag is defined so that the module is loaded in Global Script Runtime. This does not need to be the procedure in which the tag was defined.

This process is necessary because actions from Global Script and Graphics Designer are processed independently of each other in Runtime. There is no common data area between the two Runtime systems.

If you need to synchronize actions from Global Script and Graphics Designer, use the DataSet object or internal WinCC tags.

Using Global Tags in Graphics Designer

When using global tags in Graphics Designer, observe the following conditions:

- In order to use a global tag in cyclic or tag triggered action in Graphics Designer, call in the procedure in which the tag is defined. This also applies when the tag has already been called in an event triggered action.

- In order to use a global tag in an event triggered action in Graphics Designer, call in the procedure in which the tag is defined. This also applies when the tag has already been called in a cyclic or tag triggered action.

This process is necessary because the cyclic/tag triggered actions and the event triggered actions in Graphics Designer in Runtime are processed independently of each other in Runtime. There is no common data area between the two types of action.

If you need to synchronize cyclic or tag-triggered actions and event-triggered actions, use the DataSet object or internal WinCC tags.

In the case of Graphics Designer, it is also possible to define global tags in a separate declaration section. Because event-triggered and cyclic/tag-triggered actions are processed separately in Runtime, the global tags can only be jointly addressed within the event-triggered or cyclic/tag-triggered actions.
See also

Basic Principles of VBScript (Page 861)
Structure of VBScript Files (Page 92)
Creating and Editing Actions (Page 52)
Creating and Editing Procedures (Page 36)
VBScript Editors (Page 26)
Use of CrossReference (Page 22)
Actions (Page 18)
Modules and Procedures (Page 15)
1.8 VBScript Editors

1.8.1 VBScript Editors

Introduction

VBScripts can be programmed at two points in WinCC:

- In Global Script: Global Script is the central editor for VBS programming. Call it in via WinCC Explorer.

- In Graphics Designer: Graphics Designer can be used to program actions related to object properties or events related to graphic objects. The action editor in Graphics Designer is called in via the context menu in the Properties dialog of a graphic object.
Restrictions, Global Script - Graphics Designer

Graphics Designer can be used to program actions and picture-specific procedures but not global procedures valid for the entire project. However, global procedures which were programmed in Global Script can be called in.

Note
This documentation is used primarily to describe Global Script and, if necessary, makes reference to deviations in the functionality compared to Graphics Designer. A detailed description of the Graphics Designer action editor is provided in the WinCC help topic "Dynamization".

Further Information
Further information on "Dynamization" is available in the WinCC documentation.

See also
Global Script Editor (Page 27)

1.8.2 Global Script Editor

Introduction
Global procedures and actions are created and edited in the Global Script editor.

Global Script provides a similar range of functions to that of the C script editor in WinCC.

Note
A detailed description of the action editor for the creation of picture-based actions and procedures in Graphics Designer is provided under the WinCC help topic "Dynamics".

Starting Global Script
Global Script is started using the shortcut menu "Open" command in the WinCC Explorer project window.
Global Script is also automatically started when a module or action is opened by means of a double click in WinCC Explorer.

**Design of Global Script**

Global Script editor is designed in accordance with the Windows standards.

The action editor in the Graphics Designer provides a similar range of functions to that of Global Script. A description of the Graphics Designer action editor is provided in the WinCC help topic "Dynamization".

**Menu bar (1) and toolbars (2)**

All the commands required to create procedures and actions are provided in the menu bar and toolbars.

The toolbars can be displayed and hidden using the "View" > "Toolbars" command and can be moved to any position within the editor.

**Navigation window (3)**

In the navigation window you manage your procedures, modules and actions. In addition you can find here code templates which you can insert by drag&drop into your action or procedure.

A procedure can be called in another procedure or action by dragging them from the navigation window and dropping them at the relevant point in the code.
The display in the navigation window is always updated during the saving of the edited document. If you change a file, this will be displayed by a * behind the file name.

The procedures contained in a module are displayed in the navigation window underneath the module file. The Actions tab control also displays the trigger and procedures configured for an action, if necessary those directly defined in an action module.

The navigation window can still be used to:

- Create subdirectories for structuring the scripts.
- Move, copy, paste, delete and rename modules and directories directly.

The display in the navigation window can be individually configured with the "View" > "Workplace" menu commands. It is possible to select whether all file types, only script files or only syntactically correct files should be displayed. The navigation window can be shown or hidden with the "View" > "Workplace" > "Display" menu commands.

**Editing window (4)**

You write and edit your actions in the editing window. Each procedures or action will be opened in its own editing window. Several editing windows can be open at the same time.

The user is supported by in the editing window by Highlight Syntax and Intellisense. All general editor functions (e.g. Undo/Redo, Find/Replace, Copy, Paste, Cut, Font Settings, Printer Settings) remain available.

**Output window (5)**

Error messages are displayed in the output window following the syntax check. Double click on the corresponding error line to access the related point in the code.

**Status bar (6)**

The status bar contains information on the currently selected functionality or hints on programming.

---

**Note**

If information on individual editor commands or icons is required, select the “?” menu icon. > "What's This?". Then click the mouse button on the corresponding icon/command. This provides fast, direct help on all the operating elements in the editors. Press "ESC" to exit "What's This?" help mode.

---

**Window docking**

Window docking is a useful tool for the flexible arrangement of windows. It lets you reposition windows to obtain separate windows, or group windows in tab groups. For example, you can arrange your actions horizontally, vertically, or as tab group. You can automatically hide windows and show them again when needed.
See also

Deleting Actions or Procedures (Page 35)
Working with the Toolbars (Page 33)
Working in an Editing Window (Page 30)

1.8.3 Working in an Editing Window

Introduction

Procedures and actions are edited in the editing window.

Declaration Areas in Actions (Graphics Designer only)

If you create actions in the Action Editor of Graphics Designer, you can display the declaration area of the action in the editing window using the button 

The declaration area can also be used to make general settings to be used globally for the current picture, e.g.:

- Tag Definitions
- Procedures which you only want to use in this picture

Note

Do not create any directly executable codes in the declaration area!

Please note that when creating a tag, it must not contain a value (Value = VT_EMPTY). Initialize the tags after declaration with the corresponding value.

When making definitions in the declaration area, pay attention to the structure of the Script files, as described under "Structure of VBScript files".

"Option explicit" in Actions

When creating a new action, the "Option explicit" instruction is automatically set and cannot be deleted in the declaration area (Graphics Designer) or entered in the first line of an action (Global Script). The instruction is necessary as it prevents errors caused by the incorrect notation of tags without declaration. The instruction requires that tags are always defined in your code with the "Dim" instruction.

Note

Do not use the "Option explicit" instruction in the code because it may cause Runtime errors.

User Support in the Editing Window

The following functions are available to support working in the editing window.
Color coding and indentation in the editing window

Certain parts of the code have the following default colors:

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td>Key words</td>
<td>Sub, End Sub, Next</td>
</tr>
<tr>
<td></td>
<td>Functions</td>
<td></td>
</tr>
<tr>
<td>green</td>
<td>Comments</td>
<td>’ is a comment</td>
</tr>
<tr>
<td>red</td>
<td>Strings (character strings and</td>
<td>&quot;Object1&quot;</td>
</tr>
<tr>
<td></td>
<td>digits)</td>
<td></td>
</tr>
<tr>
<td>dark blue</td>
<td>Preprocessor statements</td>
<td>--</td>
</tr>
<tr>
<td>bold black</td>
<td>Constants</td>
<td>vbTrue, vbFalse</td>
</tr>
<tr>
<td>black</td>
<td>Other codes</td>
<td>--</td>
</tr>
</tbody>
</table>

The color coding in the editing window can be customized by means of the editor settings. Select the "Tools" > "Options" menu commands and the "Script Editor Options" dialog to define the settings.

In order to organize the codes clearly, they can be structured by indentations. The "Script Editor Options" dialog can also be used to define the tabulator distance and Automatic Indent while writing.

Intellisense and Highlight Syntax

During text entry, context-sensitive lists appear containing the properties, methods, and objects possible at the current code position. If you insert an element from the list, the required syntax is also indicated automatically.

Note

Full intellisense for all objects can only be utilized in the Graphics Designer if the list is accessed using the object name and the result is assigned to a tag. Otherwise, only a list of standard properties is offered.

Example of full intellisense:

```vbs
Dim Tag
Set Variable = ScreenItems ("Kreis1")
Tag.
```

If picture window limits are exceeded during addressing, it is once again only the standard properties which are offered since the picture of the picture window is not loaded.

Highlight Syntax can be activated and deactivated in the "Script Editor Options" dialog. The dialog can be called in using the "Tools" > "Options" menu commands.

General VBS Functions

Use the "Function List" command of the shortcut menu in the editing window to display a list of general VBS functions.

Lists of Objects, Properties and Methods

Using the shortcut menu in the editing window, you can view a list of the possible objects by calling the "Object List" command in Graphics Designer. Global Script only provides the
“HMIRuntime” object in this list because there is no direct access to the objects of Graphics Designer.

Use the “Properties/Methods” command of the shortcut menu to call in a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> according to the context of the script.

**Code Templates**

In the “Code templates” tab in the Navigation window of the Editor, you will find a selection of frequently used instructions, e.g., for loops and conditional instructions. The templates can be inserted in the procedure code with “drag-and-drop”.

If you want to insert a code template into your code, you have to replace the "_XYZ_" placeholder in the templates with the respective data.

**Selection Dialogs**

If WinCC tags or objects are used in the code, the following selection dialogs are available for use:

-  
  Opens a tag selection dialog and returns the selected tag name as the return value.

-  
  Opens a tag selection dialog and returns the tag name with an associated reference.

-  
  Opens a picture/object browser in which you can select a picture/object whose name is then used for the return value.

-  
  Opens a picture selection dialog for pictures and returns the picture name with the server prefix, if necessary.

**Bookmarks**

Bookmarks can be set in the code to find certain parts in the code more easily:

-  
  Sets a bookmark in the line where the cursor is currently located.

-  
  Deletes all bookmarks in the active editing window.

-  
  Jumps ahead to the next bookmark in the code.

-  
  Jumps back to the last bookmark in the code.

**See also**

-  Structure of VBScript Files (Page 92)

-  Global Script Editor (Page 27)
### 1.8.4 Working with the Toolbars

**Purpose**

In their default position, the toolbars are located below the menu bar, at the top of the VBS editor. The buttons arranged on the toolbars enable quick, easy access to the functions provided in Global Script and the action editor in Graphics Designer.

Global Script/Graphics Designer provide the following toolbars:

#### "Standard" toolbar

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
<th>Key combination</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Create Project Module" /></td>
<td>Creates a new project module (Global Script only)</td>
<td>&lt;ALT+F1&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Create Standard Module" /></td>
<td>Creates a new standard module (Global Script only)</td>
<td>&lt;ALT+F2&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Create Global Action" /></td>
<td>Creates a new global action (Global Script only)</td>
<td>&lt;ALT+F3&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Open Existing Action" /></td>
<td>Opens an existing action or existing module (Global Script only)</td>
<td>&lt;CTRL+O&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Save Editing Window" /></td>
<td>Saves the content of the active editing window. This function is only available if an editing window is open. After saving the display in the navigation window is refreshed. (Global Script only)</td>
<td>&lt;CTRL+S&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Print Content" /></td>
<td>Prints the contents of the active editing window as project documentation. This function is only available if an editing window is open.</td>
<td>&lt;CTRL+P&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Cut Text" /></td>
<td>Cuts the selected text and copies it to the clipboard. This function is only available if text has been selected.</td>
<td>&lt;CTRL+X&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Copy Text" /></td>
<td>Copies the selected text to the clipboard. This function is only available if text has been selected.</td>
<td>&lt;CTRL+C&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Paste Text" /></td>
<td>Pastes the contents of the clipboard at the location of the cursor. This function is only available if the clipboard is not empty.</td>
<td>&lt;CTRL+V&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Undo Editor Action" /></td>
<td>Undoes the last of a maximum of 30 editor actions. This function is only available if an editor action has been executed.</td>
<td>&lt;CTRL+Z&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Redo Editor Action" /></td>
<td>Redoes the last editor action that was undone. This function is only available if an editor action has been undone.</td>
<td>&lt;CTRL+Y&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Activate Direct Help" /></td>
<td>Activates the direct help</td>
<td>--</td>
</tr>
</tbody>
</table>

#### "Editor" Toolbar Content

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
<th>Key combination</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Indent Right" /></td>
<td>Indents the line, in which the cursor is located, one position to the right.</td>
<td>--</td>
</tr>
<tr>
<td><img src="image" alt="Indent Left" /></td>
<td>Indents the line, in which the cursor is located, one position to the left.</td>
<td>--</td>
</tr>
<tr>
<td><img src="image" alt="Comment Lines" /></td>
<td>Marks the lines selected by the mouse as comments. If no lines have been selected by the mouse, the line in which the cursor is located is marked as a comment.</td>
<td>--</td>
</tr>
</tbody>
</table>
### Content of the "Edit" Toolbar

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
<th>Key combination</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Button" /></td>
<td>Selects the file in the navigation window to which the current editing window belongs (Global Script only).</td>
<td>--</td>
</tr>
<tr>
<td><img src="image2" alt="Button" /></td>
<td>Displays all the files in the navigation window (Global Script only).</td>
<td>--</td>
</tr>
<tr>
<td><img src="image3" alt="Button" /></td>
<td>Only displays the Script files in the navigation window (Global Script only).</td>
<td>--</td>
</tr>
<tr>
<td><img src="image4" alt="Button" /></td>
<td>Only displays the syntactically correct files in the navigation window (Global Script only).</td>
<td>--</td>
</tr>
<tr>
<td><img src="image5" alt="Button" /></td>
<td>Executes a Syntax Check in the code of the current editing window.</td>
<td>&lt;F7&gt;</td>
</tr>
<tr>
<td><img src="image6" alt="Button" /></td>
<td>Opens a tag selection dialog returns the selected tag name as the return value.</td>
<td>&lt;CTRL+U&gt;</td>
</tr>
<tr>
<td><img src="image7" alt="Button" /></td>
<td>Opens a tag selection dialog and returns the tag name with an associated reference.</td>
<td>&lt;CTRL+W&gt;</td>
</tr>
<tr>
<td><img src="image8" alt="Button" /></td>
<td>Opens a picture/object browser in which a picture/object can be selected whose name is then used for the return value.</td>
<td>&lt;CTRL+Q&gt;</td>
</tr>
<tr>
<td><img src="image9" alt="Button" /></td>
<td>Opens a picture selection dialog for pictures and returns the picture name, with the server prefix if necessary.</td>
<td>&lt;CTRL+B&gt;</td>
</tr>
<tr>
<td><img src="image10" alt="Button" /></td>
<td>Opens the &quot;Info/Trigger&quot; dialog.</td>
<td>&lt;CTRL+T&gt;</td>
</tr>
</tbody>
</table>

### Additional Buttons in Graphics Designer

In addition to the buttons provided by Global Script, the action editor in Graphics Designer also has the following buttons:

- Displaying the declaration area (<CTRL+E>)
- Hiding the declaration area (<CTRL+A>)
See also

Global Script Editor (Page 27)

1.8.5 Deleting Actions or Procedures

Introduction

If an action, procedure or a module is deleted in a script editor, the code and corresponding file are deleted in the project directory.

Be careful only to delete procedures which are no longer used in other procedures or actions. If an action attempts to call in a procedure which no longer exists, the action is stopped in Runtime at the fault point. A non-existing reference in the code is not detected by the syntax check.

Note

Procedures can only be deleted within a module by deleting the code, not in the editor's navigation window.

Procedure

1. Open Global Script.
2. Select the action or module to be deleted in the navigation window.
3. Select the "Delete" command from the context menu.
4. To delete a procedure: Open the relevant module and delete the corresponding code in the editing window.

See also

Actions (Page 18)
Modules and Procedures (Page 15)
Global Script Editor (Page 27)
1.9 Creating and Editing Procedures

1.9.1 Creating and Editing Procedures

Introduction

Projects and standard procedures can be programmed in WinCC using VBS:

- Project procedures can only be retrieved in the current project. Since procedures are stored in the project directory, they are automatically copied when a project is copied.

- Standard procedures can be called in by all computers linked to a project. When a project is copied onto another computer, the standard procedures must be copied into the corresponding directory on the target computer manually.

The copied procedures are directly available for use in Runtime. They become visible in the editor when the view is updated.

Apart from the procedures programmed by yourself, general VBS functions can also be used (e.g. Abs, Array,... Year). These general VBS functions can be invoked in the code using the "Function List" command from the context menu.

In addition, WinCC provides the most popular instructions as code templates (e.g. If...Then, When...While). The code templates can be moved from the navigation window's Code Templates tab control directly into your code via drag&drop.

If you insert a code template into your code, it is important to note that, for example, conditions in the templates are identified by "_XYZ_". You must replace these placeholders with the appropriate information.

Using Procedures

Procedures are used for the central creation and maintenance of codes which are to be implemented at several points in the configuration. Codes are written and saved in a procedure and the procedure is called in with the current parameters in actions or other procedures instead of repeatedly entering the same code.

Create procedures for repeated functionalities, e.g.

- Calculations with different starting values (procedure with return value)
- Checking tag values (procedure with return value)
- Executing tasks (procedure with return value)

This is linked to the following advantages:

- The code is only programmed once.
- Modifications are only made at one point, namely in the procedure, not in each action.
- The action code is shorter and, thus, remains clearer.

Related procedures should be stored in modules in WinCC.
Procedures are loaded in Runtime when the calling action is executed.

If a procedure (module) used in a picture is modified, the modification is applied the next time the picture is loaded. This means that a picture currently displayed only works with the modified procedure after the picture has been reloaded.

After having a changed project module and saved the file in the VBS Editor, you must also open and save the corresponding process picture in Graphics Designer. You changes are not
activated in Runtime unless you completed this action. By saving the picture, you activate the information in the picture file by means of the necessary project modules.

Note

Procedures can be used in actions in Global Script and Graphics Designer. In order to use a global tag defined in Global Script in an action in Graphics Designer, observe the following:

In order that access can be made to the tag, it is necessary to call in the procedure in which the tag is defined.

In order to use a global tag in picture-independent actions in Global Script, observe the following:

In order that access can be made to the tag, at least one procedure in the module containing the tag must be called in at least one global action.

Procedure - Action Restrictions

Global procedures valid for the entire project can only be created in Global Script. Graphics Designer can only be used to create picture-specific procedures and call in global procedures in actions. Picture-specific procedures in Graphics Designer are defined in the declaration area of an action.

A procedure is not executed without an action.

File Name and Procedure Name

The procedure name is entered in the first line of the procedure code. The procedure is displayed in the navigation window and called in actions under this name. Procedures do not have a file name of their own but are stored in a module.

Module names are assigned in the editor’s navigation window. Use the “Save As” command to save a module under another name in the project directory.

Since procedures in Global Script are valid for the entire project, procedure names must always be unique. Module names can be used more than once within a project, e.g. in different subdirectories or stored separately in the standard and project directories.

Displaying Procedures and Modules

If you save a module that contains at least one syntactically incorrect procedure, this will be displayed in the navigation window with this adjacent symbol.

Note

If a module contains a syntactically incorrect procedure, the module can no longer be loaded. Procedures can no longer be called from the module.

If you save a module that contains only syntactically incorrect procedures, this will be displayed in the navigation window with this adjacent symbol.
Procedures and Modules

Procedures are classified as standard or project procedures according to their assignment to standard or project modules. Standard and project modules are located on the corresponding tab controls in the Global Script navigation window.

Use the modules in order to compile procedures to practical function groups. Observe the following when structuring the modules and procedures:

- The more modules that must be loaded when a picture is called in, the worse the performance in Runtime.
- The larger a module, the more procedures are contained and, thus, the longer the loading time for the module.

Organize the modules sensibly, e.g. a module with procedures for a specific system part/picture.

See also

- Multiple Use of Procedures and Actions (Page 20)
- Renaming a Procedure or Module (Page 50)
- Saving a Procedure (Page 48)
- Protecting a Module with a Password (Page 47)
- How to add module-related information (Page 45)
- Using Standard and Project Procedures (Page 44)
- How to Write Procedure Codes (Page 41)
- Creating a New Procedure (Page 39)
- Examples of VBScript (Page 823)
- Modules and Procedures (Page 15)

1.9.2 Creating a New Procedure

Introduction

Standard projects and procedures can be programmed with Global Script in WinCC.

The type of procedure is defined by the assignment to a project or standard module. The procedure to create standard or project procedures is identical.

On creating a new procedure, WinCC automatically assigns a standard name "procedure#", in which case # represents a continuous number. If the procedure is edited in the editing window,
assign the procedure a corresponding name via which the procedure can be called in an action later. The name appears in the navigation window when the procedure is saved.

**Note**

Procedure names must be unique within a project. If a procedure with the same name already exists, the module is identified as syntactically incorrect. Module names can be used twice when the modules are stored in different directories.

Global procedures (valid for the entire project) can only be programmed in Global Script. Procedures can be called via actions in Graphics Designer and picture-related procedures created in the declaration area of an action. Using a global action in Global Script, it is possible to create procedures directly in the code which are then only applicable for this action.

An action must be programmed in order to call in a procedure.

**Procedure**

1. Open Global Script.
2. In the navigation window, select the standard modules or project modules tab control according to whether a standard procedure or project procedure is to be created.
3. Open an existing module or create a new module using the menu commands “File” > “New” > “Project Module” or “File” > “New” > “Standard Module”.
4. After having created a new module, the structure of a procedure without return value is already entered in the editing window:
5. Enter the procedure name direct in the code: `Sub "Procedure name"`.
6. To insert one procedure in an existing module: Select the module in the navigation window and then select the “Insert New Procedure” context menu item.
7. Enter a procedure name and select whether the procedure should have a return value parameter. The definition of a tag for the return value is then entered in the code (Dim RetVal).
8. Confirm your settings with OK.

**Note**

A new procedure can also be entered directly in a module. In the case of procedures without return value, always begin with the instruction “Sub ” and <Procedure Name> and conclude with ”End Sub”. In the case of procedures with return value, always begin with the instruction “Function ” and <Procedure Name> and conclude with ”End Function”. The new procedure is displayed in the navigation window when the module is saved.
See also

- Creating and Editing Procedures (Page 36)
- Multiple Use of Procedures and Actions (Page 20)
- Renaming a Procedure or Module (Page 50)
- Saving a Procedure (Page 48)
- Protecting a Module with a Password (Page 47)
- How to add module-related information (Page 45)
- Using Standard and Project Procedures (Page 44)
- How to Write Procedure Codes (Page 41)
- Modules and Procedures (Page 15)

1.9.3 How to Write Procedure Codes

Introduction

Procedure codes are written in the Global Script editor window. The code of each procedure can call in other procedures via their procedure names.

Procedures can be created with or without return values. Use the return value to receive information about successful execution of the procedure, for example.

If you modify a procedure in a picture, the modification will not take effect until the next time you load the picture.

Functions in Global Script

Global Script provides the following functions to support the creation of procedure codes:
Intellisense and Highlight Syntax
During text entry, context-sensitive lists appear containing the properties, methods, and objects possible at the current code position. If you insert an element from the list, the required syntax is also entered automatically.

Note
Full intellisense for all objects can only be utilized in the Graphics Designer if the list is accessed using the object name and the result is assigned to a tag. Otherwise, you are only offered a list of standard properties.

Example of a full intellisense:
Dim Variable
Set Variable = ScreenItems ("Circle1")
Variable.<<Intellisense selection>>

If picture window limits are exceeded during addressing, it is once again only the standard properties which are offered since the picture of the picture window is not loaded.

General VBS Functions
Use the "Function List" command of the shortcut menu in the editing window to display a list of general VBS functions.

Lists of Objects, Properties and Methods
Using the shortcut menu in the editing window, you can view a list of the possible objects by calling the "Object List" command in Graphics Designer. Global Script provides only the "HMIRuntime" object in this list because there is no direct access to the objects of Graphics Designer.

Use the "Properties/Methods" command of the shortcut menu to call a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> according to the context of the script.

Code Templates
In the "Code templates" tab in the Navigation window of the Editor, you will find a selection of frequently used instructions, e.g. for loops and conditional instructions. The templates can be inserted in the procedure code with "drag-and-drop".

If you want to insert a code template into your code, you have to replace the "_XYZ_" placeholder in the templates with the respective data.

Selection Dialogs
If WinCC tags or WinCC objects are used in the code, the following selection dialogs are available for use:

- Opens a tag selection dialog and returns the selected tag name as the return value.
- Opens a tag selection dialog and returns the tag name with an associated reference.
● Opens a picture/object browser in which a picture/object can be selected whose name is then used for the return value.

● Opens a picture selection dialog for pictures and returns the picture name, with the server prefix if necessary.

Syntax Check
Global Script supports you with a syntax check which you can perform after the code has been created. Syntax errors in the code are displayed in the output window of the editor. You can move to the erroneous point in the code by double-clicking the error in the output window.

Note
The syntax check can only detect syntax errors in the code. Programming errors, such as missing references, only become visible in Runtime. You should therefore also always check your scripts in the Runtime environment.

Changing a Procedure
If a procedure is modified during Runtime, the modification becomes active at the following times:

● Procedures called from actions or other procedures in pictures become active following a picture change.

● Procedures in Global Script become active directly after being called again.

Procedure
1. Open Global Script.
2. Open the module containing the procedure to be edited.
3. After double clicking on the procedure in the navigation window, the cursor skips to the beginning of the required procedure.
4. Edit the procedure. If you create a procedure with a return parameter, e.g. to program recurring evaluations or reviews, indicate the return value with "procedurename = RetVal" at the end of the procedure.

See also
How to Write Procedure Codes (Page 41)
Multiple Use of Procedures and Actions (Page 20)
Renaming a Procedure or Module (Page 50)
Saving a Procedure (Page 48)
Protecting a Module with a Password (Page 47)
How to add module-related information (Page 45)
Creating a New Procedure (Page 39)
1.9.4 Using Standard and Project Procedures

Introduction

Use the drag&drop function in the navigation window or the context menu to insert a procedure in the current code.

Project procedures can only be used within the current project, standard procedures are available for all projects on the computer.

Procedures, once created, can be used in other projects or on other computers. To do this, copy the module containing the procedures in the appropriate project or standard directory.

Using Procedures in Graphics Designer and Global Script

Procedures defined in Global Script can be called in actions in Global Script and Graphics Designer. On executing the action in Runtime, the entire module which contains the procedure is loaded.

Please observe the following in order to use a global tag defined in a procedure in Global Script:

When using Graphics Designer, always call the procedure in which the tag is defined so that the tag can be used. If the procedure is not called in, the corresponding module is not loaded and access cannot be made to the tag.

In the case of picture-independent actions in Global Script, at least one procedure in the module containing the tag must be called in at least one global action.

Note

In the general declaration part of screens, no check is made whether a procedure or function name has already been assigned. Therefore, a name could occur several times and it is not defined which function will be executed. This is standard behavior of the MS Scripting Engine.

Procedure

1. Open the procedure or action in which the procedure should be inserted.
2. Use the drag&drop function to move the procedure to be inserted from the navigation window to the correct position in the code.
   or
3. Place the cursor at the position in the code where you would like to insert the procedure.
4. Mark the procedure in the navigation window with the mouse.
5. Choose pop-up menu command "Transfer Procedure Retrieval".
1.9.5 How to add module-related information

Introduction

Related information can be added to each module in order to quickly recognize the functionality of the module or the procedures contained in it when edited at a later date. If several operators are involved in configuring a project, you should provide module-related information for your colleagues.

When a new module is created, the creation date is entered in the module-related information automatically and is unchangeable. The Module is also assigned the version number 1.0. The version numbers can be individually assigned when editing a module. When a module is changed and saved, the current date of change is entered automatically and is unchangeable.

The following information can be added:

- "Created by"
- "Changed by"
- "Comments" e.g. module functionality/procedures contained

It is still possible to define a password for the module. Further information on assigning passwords is provided in "Protecting Modules with a Password".
Procedure

1. Open Global Script.
2. Select the module in which information is to be added in the navigation window.
3. Click the "Info/Trigger" toolbar button, or select the "Info" menu command. The "Properties..." dialog appears.

![Properties dialog]

Enter the required information.

Note

The "Info/Trigger" dialog can also be called if an open procedure is selected in the navigation window. The information stored in this dialog is always valid for the entire module and all the procedures contained in it.

See also

Multiple Use of Procedures and Actions (Page 20)
Renaming a Procedure or Module (Page 50)
Saving a Procedure (Page 48)
How to add module-related information (Page 45)
Using Standard and Project Procedures (Page 44)
How to Write Procedure Codes (Page 41)
Creating a New Procedure (Page 39)
1.9.6 Protecting a Module with a Password

Introduction

A module can be assigned a password to protect it from unauthorized access. The password is a part of the module-related information.

Note

If a module is protected by a password, all the procedures contained in it are also protected by the password.

Procedure

1. Open Global Script.
2. Select the module to be assigned a password in the navigation window.
3. Click the button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info". The "Properties..." dialog appears.
4. Activate the check box "Password". The dialog "Enter Password" is displayed.
5. Enter a password and confirm it.
6. Confirm your settings with OK.

Result

If an attempt is made to open the module or a procedure contained in it, a prompt appears requesting the password.

Deactivate Password Protection

To clear the password protection, disable the "Password" check box.

Change Password

To change the password, open in the Properties dialog and click the "Change" button. Then enter the new password.

Note

If you forget the module password, the module cannot be edited.
Note

The "Info/Trigger" dialog can also be called if an open procedure is selected in the navigation window. The information stored in this dialog is always valid for the entire module and all the procedures contained in it.

See also

- Multiple Use of Procedures and Actions (Page 20)
- Renaming a Procedure or Module (Page 50)
- Saving a Procedure (Page 48)
- How to add module-related information (Page 45)
- Using Standard and Project Procedures (Page 44)
- How to Write Procedure Codes (Page 41)
- Creating a New Procedure (Page 39)
- Modules and Procedures (Page 15)
- Creating and Editing Procedures (Page 36)

1.9.7 Saving a Procedure

Introduction

Individual procedures are never stored but the module in which the procedure has been programmed.

Before saving a module, check the code is syntactically correct. When saving a module, the procedures contained are automatically checked and, in the case of syntax errors, a prompt appears as to whether the module should be saved with the errors or not. In this way, for example, modules and procedures can be saved which are not fully programmed. Syntactically incorrect procedures do not run in Runtime.

Note

If a module contains a syntactically incorrect procedure, the module can no longer be loaded. Procedures can no longer be called from the module.
**Note**

The syntax check can only detect syntax errors in the code. Programming errors, such as missing references, only become visible in Runtime. Therefore, always check the scripts in the Runtime environment and use a debugger, if necessary, to detect and eliminate errors.

Only syntactically correct modules are called in Runtime.

A list of all the possible syntax errors is available in the Appendix under "Basic Principles of VBScript".

---

If a procedure is subjected to a syntax check prior to saving, any errors are displayed in the lower part of the editor window. Double click on an error line to access the error position in the code directly.

---

Use the "Save As" command to store the module under another name. Note that the new module is only displayed in the navigation window after updating the view.

---

**Requirement**

The procedure/module to be saved must be open in the editor window.

---

**Procedure**

1. Click in the toolbar.
2. If syntax errors appear in the output window, double click on the error line and correct the error in the code. Repeat steps 1 and 2 until the code is correct.
3. Save the module by clicking in the toolbar.

---

**Note**

*Pictures with modified procedures must be opened and saved once again in Graphics Designer*

In addition to saving in the VBS editor, the corresponding picture must be opened and saved once again in Graphics Designer when the project module is changed. The change is then applied in Runtime. Only once the picture has been saved is the information applied to the picture file via the required project modules.

---

**See also**

Diagnostics (Page 80)

Multiple Use of Procedures and Actions (Page 20)
### 1.9.8 Renaming a Procedure or Module

**Introduction**

Procedures and modules are renamed in the following cases:

- When a standard name (procedure# or Modul#), which was automatically assigned when the new module/new procedure was created, is changed to a self-explanatory name.

- When a module or procedure is copied in order, for example, to create a new module with similar content from an existing one. Please note that procedure names must be unique within a project. Procedure names which exist twice are issued as errors during the syntax check. Contrary to procedure names, the same name can be applied to modules when the modules are stored in different directories.

**Note**

The module name is always identical to the file name in the WinCC file system. If a module name is changed, e.g. in Windows Explorer, the new module name is taken over from Global Script in the navigation window.

**Procedure**

**Renaming Procedures**

1. Open the procedure to be renamed.

2. Enter the new name in the header of the procedure.

3. Save the procedure so that the name is transferred to the navigation window. Procedure names are always unique and may not be used more than once.
Renaming Modules

1. Close the module to be renamed.

2. Select the module in the navigation window and choose the "Rename" option from the context menu.

3. Enter the new name in the navigation window. Module names are always unique at directory level and may not be used more than once.

See also

Multiple Use of Procedures and Actions (Page 20)
Saving a Procedure (Page 48)
Protecting a Module with a Password (Page 47)
How to add module-related information (Page 45)
Using Standard and Project Procedures (Page 44)
How to Write Procedure Codes (Page 41)
Creating a New Procedure (Page 39)
Modules and Procedures (Page 15)
Creating and Editing Procedures (Page 36)
1.10 Creating and Editing Actions

1.10.1 Creating and Editing Actions

Introduction

When using VBS in WinCC, there is no differentiation between local (valid for entire project) and global (valid on all computers) actions, as opposed to C. A configured action is always valid globally.

A copied action is available for use in Runtime following a restart or opening and saving the action. The become visible in the editor when the view is updated.

VBS actions can be used to make graphic objects and object properties dynamic in Runtime or to execute picture-independent actions.

Note

Please note that the object name length of objects made dynamic in Graphics Designer is limited to approx. 200 characters, and each special character used in an object name is converted to five characters in the script files. The special characters are represented by a four-place hexadecimal code behind the preceding X. If the name of an object made dynamic is too long, a corresponding error message appears. Further information is available in this help under "Structure of VBScript Files".

Note

If you make an object property dynamic with a VBS action via the return value of a script, the value of the object property is written only if it has changed in relation to the last script run. It is not considered if the value had been changed from another location.

Therefore it is illegal to change properties which have been made dynamic by VBS action via the return value from another location (e.g., other C scripts or VBS scripts).

if you do not observe this, wrong values can be the results.

Using the Actions

Actions can be used as follows:

On graphic objects in Graphics Designer

Making properties dynamic (action with return value), e.g.:

Function BackColor_Trigger(ByVal Item)

'VBS143

    BackColor_Trigger = RGB(125,0,0)

End Function

Triggered by an event on an object (action without return value), e.g.:
Sub OnClick(ByVal Item)
    'VBS144
    Item.BackColor = RGB(255,0,0)
End Sub

**Picture-independent in Global Script**

As a cyclic action, e.g. incrementing a tag:

Option Explicit

Function action
    'VBS145
    Dim objTag1
    Dim lngValue
    Set objTag1 = HMIRuntime.Tags("Tag1")
    lngValue = objTag1.Read
    objTag1.Write lngValue + 1
    action = CLng(objTag1.value)
End Function

**Executing Actions**

An action can be assigned several triggers. The action is always executed when one of the
triggering events occurs. Observe the following:

- Actions in Global Script cannot be executed simultaneously. The action triggered last is
  held in a queue until the action currently being performed is completed.

- When using Graphics Designer, cyclically and tag-driven actions cannot be triggered
  simultaneously. If the execution of a tag-driven action hinders the execution of a cyclic
  action, the cyclic action is executed when the tag-driven action has finished. The cyclic
  action is held in a queue during the non-execution phase. When the current action is
  completed, the cyclic action is executed with the normal cycle.

- In Graphics Designer, event-driven actions cannot be executed simultaneously.

The action types mentioned do not prevent each other being executed: The execution of
actions in Global Script has no influence on actions in Graphics Designer. In the same way,
in Graphics Designer, the execution of cyclically or tag-driven actions has no effect on the
execution of event-driven actions.

**Note**

Actions in pictures which are still running one minute after the picture has been deselected
are terminated by the system. This is recorded in a logfile entry.
Locating Actions

All the actions used in a picture can be displayed by means of the picture properties. To do this mark the picture in WinCC Explorer and select the "Properties" context menu command. After double clicking on an entry, detailed information on the type of dynamics appears.

It is also possible to display all the tags and pictures used in actions by means of the WinCC CrossReference. CrossReference can also be used for the to link tag connections of Graphics Designer actions easily.

Note

Use the standard formulations
 HMIRuntime.BaseScreenName = "Screenname" and HMIRuntime.Tags("Tagname") when addressing pictures and tags in your code in order to ensure that the pictures and tags are registered by the CrossReference.

Procedure - Action Restrictions

Actions can be used to program instructions and call procedures. Codes are programmed within in procedures for use at several points in a configuration. Contrary to procedures, actions always have a trigger.

Creating and Editing Actions

Actions can be configured in Global Script and Graphics Designer. Use Global Script to configure global actions which can be executed independently of the picture currently open. Graphics Designer is used to configure actions related to graphic objects which should be executed when the picture is opened in Runtime or when the configured trigger occurs.
The script editors in WinCC provide the option of checking that scripts have a correct syntax without executing them. Errors in the script are displayed in the output window under the editor window. Double click on the corresponding error line to access the related point in the code.

Note
The syntax check can only check objects known to the system at the moment of execution. Therefore, the tags and objects addressed in the scripts must be created in WinCC. Only syntactically correct actions are executed in Runtime. The automation objects "PDLRuntime" and "WinCC Runtime Project" cannot be used in VBS actions.

Display of Actions
- If you save a syntactically incorrect action, it will be displayed in the navigation window with this adjacent symbol.
- If you save a syntactically correct action without trigger, it will be displayed in the Global Script navigation window with this adjacent symbol.
- If you save a syntactically correct action with trigger, it will be displayed in the Global Script navigation window with this adjacent symbol.

Note
Actions can only be saved in the Graphics Designer if they have the correct syntax. If an action with errors should still be retained and exited, however, enter the comments.

System behavior if actions are changed, deleted and saved at Runtime
If a local action is stored at runtime, then all local and global actions of the computer are reset on the computer to which the local action belongs.
If a global action is stored during runtime, then all local and global actions for the entire project – and thus on all computers – are reset.
Such a reset might reinitialize for examples tags and times that are used as triggers for actions, triggering the action at that stage.
Static tags used in the reset actions are reinitialized.

See also
- Structure of VBScript Files (Page 92)
- Action and Procedure Names in the Debugger (Page 94)
- How to Rename an Action (Page 76)
- Saving Actions (Page 63)
- Protecting an Action with a Password (Page 62)
1.10.2 Creating a New Action

Introduction

When a new action is created, the editor automatically suggests a file name (Action#.bac), which can be changed.

Actions can be configured in Global Script and Graphics Designer.

- Global Script is used to configure actions which can be executed, picture-independently, in Runtime. Open Global Script via WinCC Explorer.
- Graphics Designer is used to configure a new action, related to the properties of a graphic object, by clicking on the right mouse button in the "Dynamic" column of the Properties tab control and selecting VBS Action. An action, related to an event, is created in the same way using the Events tab control.

In both cases, the Action Editor of the Graphics Designer opens.

Note

The precise procedure for linking actions with graphic objects is described under the WinCC help topic "Dynamics".

Procedure

1. Open Global Script.
2. Activating the Actions Tab Control in the Navigation Window.
3. Click in the toolbar or choose the menu command "File" > "New" > "Action".
   A new action is opened in the editor window. The action appears in the navigation window after it has been saved.

Note

When creating a new action, the "Option explicit" instruction is automatically entered in the declaration area and cannot be deleted. The instruction is necessary as it prevents errors caused by the incorrect notation of tags without declaration.

The instruction requires that tags are always defined in your code with the "Dim" instruction.

Do not use the "Option explicit" instruction in the code because it may cause Runtime errors.
See also

How to Rename an Action (Page 76)
Saving Actions (Page 63)
Protecting an Action with a Password (Page 62)
How to add action-related information (Page 60)
How to Edit Actions (Page 57)
Triggers (Page 64)
Creating and Editing Actions (Page 52)
Actions (Page 18)

1.10.3 How to Edit Actions

Introduction

An action is edited in the same way as a procedure in the editor window of the editor or in the Graphics Designer action editor.

In order that an action can be executed in Runtime, it requires a trigger. Actions which are triggered by an event in Graphics Designer do not require the assignment of a trigger.

If an action is modified during Runtime, the change is applied when the picture is reloaded (in the case of actions in Graphics Designer) or the next time the action is called (in the case of actions in Global Script).

---

Note

A change in the code in Runtime cannot be applied when another action is being carried out at the same time.

A procedure call can be inserted in the action by dragging the procedure from the editor navigation window with "drag-and-drop" and dropping it in the corresponding position of the code in the editor window. C scripts cannot be called in VBS actions.

Declaration Area in Actions

If you create actions in Graphics Designer, you can display the declaration area of the action using the button 📝. When creating a new action, the "Option explicit" instruction is automatically entered in the declaration area and cannot be deleted. The instruction is necessary as it prevents errors caused by the incorrect notation of tags without declaration.

The instruction requires that tags are always defined in your code with the "Dim" instruction.

Do not use the instruction "Option explicit" in your code as this can cause Runtime errors.
In the declaration area, you can also make general settings which you want to use globally for the current picture, e.g.:

- Tag Definitions
- Procedures which you only want to use in this picture

In the declaration area of the actions, you may define global tags independent of each other in the areas "Event" and "Properties" of an object. There is no link between global tags of identical names in both areas.

**Note**

Always make sure that the procedures in the declaration area have correct syntax, i.e. with "Sub" - "End Sub". Do not create directly executable codes in the declaration area as this can cause Runtime errors.

If global tags are used in the declaration area of actions Graphics Designer, note that the event-driven and cyclic/tag-driven actions are processed separately in Runtime. There is no synchronization of global tags between the two Runtime systems in Runtime. If synchronization of tags is required, configure these using the DataSet object or internal WinCC tags.

When making definitions in the declaration area, pay attention to the structure of the Script files, as described under "Structure of VBScript files".

**Functions for Editing Actions**

The script editors provide the following functions to assist you in creating action code:

**Intellisense and Highlight Syntax**

During text entry, context-sensitive lists appear containing the properties, methods, and objects possible at the current code position. If you insert an element from the list, the required syntax is also indicated automatically.

**Note**

Full intellisense for all objects can only be utilized in the Graphics Designer if the list is accessed using the object name and the result is assigned to a tag. Otherwise, only a list of standard properties is offered.

Example of full intellisense:

```
Dim Variable
Set Variable = ScreenItems("Circle1")
```

If picture window limits are exceeded during addressing, it is once again only the standard properties which are offered since the picture of the picture window is not loaded.

**General VBS Functions**

Use the "Function List" command of the shortcut menu in the editing window to display a list of general VBS functions.
Lists of Objects, Properties and Methods

Using the shortcut menu in the editing window, you can view a list of the possible objects by calling the "Object List" command in Graphics Designer. Global Script only provides the "HMIRuntime" object in this list because there is no direct access to the objects of Graphics Designer.

Use the "Properties/Methods" command of the shortcut menu to call in a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> according to the context of the script.

Code Templates

In the "Code templates" tab in the Navigation window of the Editor, you will find a selection of frequently used instructions, e.g., for loops and conditional instructions. The templates can be inserted in the procedure code with "drag-and-drop".

If you want to insert a code template into your code, you have to replace the "_XYZ_" placeholder in the templates with the respective data.

Selection Dialogs

If WinCC tags or WinCC objects are used in the code, the following selection dialogs are available for use:

- Opens a tag selection dialog and returns the selected tag name as the return value.
- Opens a tag selection dialog and returns the tag name with an associated reference.
- Opens a picture/object browser in which a picture/object can be selected whose name is then used for the return value.
- Opens a picture selection dialog for pictures and returns the picture name with the server prefix, if necessary.

Syntax Check

Global Script supports you by providing a syntax check which you can perform after the code has been created. Syntax errors in the code are displayed in the output window of the editor. You can move to the erroneous point in the code directly by double-clicking the error in the output window.

Note

The syntax check can only detect syntax errors in the code. Programming errors, such as missing references, only become visible in Runtime. Therefore, always check the scripts in the Runtime environment and use a debugger, if necessary, to detect and eliminate errors. The way to test scripts with a debugger is described in this documentation under the topics "Diagnostics" > "Testing with the Debugger".
Procedure

1. Open Global Script.
2. Double click on the action on the Action tab control in the navigation window.
3. Edit the action.

See also

Using Global Tags in VBS (Page 24)
Testing with the Debugger (Page 87)
Structure of VBScript Files (Page 92)
How to Rename an Action (Page 76)
Saving Actions (Page 63)
Protecting an Action with a Password (Page 62)
How to Edit Actions (Page 57)
Creating a New Action (Page 56)
Triggers (Page 64)
Creating and Editing Actions (Page 52)
Actions (Page 18)

1.10.4 How to add action-related information

Introduction

Related information can be added to every action in Global Script in order that the function of an action can be recognized at a later date when editing. If several operators are involved in configuring a project, you should provide action-related information for your colleagues.

When a new action is created, the creation date is entered in the action-related information automatically and is unchangeable. The action is also assigned version number 1.0. The version numbers can be individually assigned when editing an action. When an action is changed and saved, the current date of change is entered automatically and is unchangeable.

The following information can be added:

- "Created by"
- "Changed by"
- "Comments:" e.g. functionality of the action
It is also possible to define a password for the action. Further information on assigning passwords is provided in "Protecting Actions with a Password".

Note
Additional information can only be made available actions in Global Script, not for actions in Graphics Designer.

Procedure

1. Open Global Script.
2. Open the action for which information should be added.
3. Click the "Info/Trigger" toolbar button, or select the "Info" menu command. The "Properties..." dialog appears.

4. Enter your information.

See also

How to Rename an Action (Page 76)
Saving Actions (Page 63)
Protecting an Action with a Password (Page 62)
How to Edit Actions (Page 57)
Creating a New Action (Page 56)
Triggers (Page 64)
1.10.5 Protecting an Action with a Password

Introduction

An action in Global Script can be protected against unauthorized access by assigning a password to it. The password is a part of the action-related information.

Note

Only actions in Global Script can be assigned a password, not actions in Graphics Designer.

Procedure

1. Open Global Script.
2. Open the action to be protected by a password.
3. Click the button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info". The "Properties..." dialog appears.
4. Select the "Password" check box.
5. Click the "Change" button. The "Enter Password" window opens.
6. Enter a password and confirm it.
7. Confirm your settings with OK.

Result

If an attempt is made to open the action, the system requests the password is entered.

Deactivate Password Protection

To clear the password protection, disable the "Password" check box.

Change Password

To change the password, open in the Properties dialog and click the "Change" button. Then enter the new password.

Note

If you forget the action password, the action cannot be edited.


1.10.6 Saving Actions

Introduction

Before an action can be run in Runtime, it must be saved. Save an action as any other Windows file using the "File" > "Save" commands or the corresponding icon.

Note

Actions in Graphics Designer are automatically applied on closing the action editor with the picture. Functions can only be saved in the Graphics Designer if they have the correct syntax. If an action with errors should still be retained and exited, however, enter the comments. A list of all the possible syntax errors is available in the Appendix under "Basic Principles of VBScript".

In order to save an action under a different name, e.g. to use an action as a basis for another action, use the "Save As" command.

Note that, when using "Save As", only the file name is changed and not the action name.

Prior to Saving

Before saving an action, check the code is syntactically correct. The syntax errors in the code are displayed in the output window of Global Script. Double click on an error line to access the error position in the code directly.

Note

The syntax check can only detect syntax errors in the code. Programming errors, such as missing references, only become visible in Runtime. Therefore, always check the scripts in the Runtime environment and use a debugger, if necessary, to detect and eliminate errors.

If actions are saved without running a syntax check beforehand, the editor comments that a syntactically incorrect action will be saved which cannot subsequently be run in Runtime.
Syntactically incorrect actions are displayed with the adjacent icon in the navigation window.

Procedure

1. Click in the toolbar.
2. If errors are displayed in the lower part of the editor window, double click on the error line and correct the error in the code. Repeat steps 1 and 2 until the code is correct.
3. Save the action by clicking in the toolbar.

See also

Actions (Page 18)
How to Rename an Action (Page 76)
Protecting an Action with a Password (Page 62)
How to add action-related information (Page 60)
How to Edit Actions (Page 57)
Creating a New Action (Page 56)
Triggers (Page 64)
Creating and Editing Actions (Page 52)

1.10.7 Triggers

1.10.7.1 Triggers

Definition and use

Triggers are used to execute actions at Runtime. To do this, a trigger is linked to an action, forming the triggering event for calling the action. Actions without triggers will not be carried out.

The triggers defined for an action are displayed in the Global Script navigation window.
Trigger types

The following trigger types are available:

**Acyclic triggers**
They consist of the specification of date and time. The action specified by such a trigger is performed once at the date and time specified.

**Cyclic triggers**
They consist of the specification of a time interval and start time. The following types of cyclic triggers are available:

- Default cycle. The start of the first time interval coincides with the start of Runtime. The length of the interval is determined by the cycle.
- Hourly. The start of the interval is specified as minute and second. The length of the interval is an hour.
- Daily. The start of the interval is specified by the time (hour, minute and second) festgelegt. The length of the interval is a day.
- Weekly. The start of the interval is specified by the day of the week (Monday, Tuesday, etc.) and the time. The length of the interval is a week.
- Monthly. The start of the interval is specified by the day and time. The length of the interval is a month.
- Annual. The start of the interval is specified by the day, month and time. The length of the interval is a year.

Time-controlled triggers are used for actions Global Script and for actions to make graphic objects dynamic.

**Tag triggers**
They consist of one or more specified tags. The action associated with such a trigger is performed each time a change in the value of one of these tags is detected.

How the tag values are queried may be customized for each tag. Select from the following modes:

- Cyclic query of the tag value: Specify a standard cycle. The tag value is queried at the defined intervals (e.g. every 2 seconds). The action is triggered when the system detects a change of the tag value. Depending on the size of the cycle, it is possible that the tag value is changed but it is not detected by the system. If, for example, a cycle of 5 minutes has been set, the tag value may change several times within the 5 minute period but only the value set when the next query is made is detected. The value changes between the two queries are not registered.
- Changes in the tag value: Each change in the tag value is detected by the system. The action is executed each time the tag value changes.

Tag triggers are used for actions Global Script and for actions to make graphic objects dynamic.
Event-driven
When an action is configured related to an event on a graphic object, the action is triggered when a specific event has occurred, e.g. following a mouse click or the change of the background color due to another action.

Animation cycle
As of WinCC V7.0, the "animation cycle" trigger art is available for the dynamization of objects with VBS. The animation cycle allows you to switch actions on and off in Runtime and to change the time, in which the trigger is executed.
You can find additional information in the "AUTOHOTSPOT" section.

Effects of triggers on actions
If the action is associated with only one trigger, then the action is performed as soon as the triggering event occurs.
However, an action may be associated with multiple triggers, such as a cyclic trigger and a tag trigger. Here the action is performed whenever one of the two triggering events occurs. If two events occur simultaneously, then the action is executed twice sequentially. If two tag triggers fire at the same time, the action will be performed only once.

Processing actions in Graphics Designer
The following rules apply to processing actions in Graphics Designer:
- No event-driven actions can be executed as long as another event-driven action is running.
- No cyclic/tag triggered actions can be executed as long as another cyclic/tag triggered action is running.
- The two action types do not affect each other: An event-driven action can also be executed when a cyclic action is already in progress.
- If the execution of actions is blocked by other actions (e.g. a cyclic action by a tag-triggered action), each action which is blocked is executed once at the next possible moment. Cyclic actions then run in their normal intervals after the one-off execution.

Processing actions in Global Script
Picture-based actions from Global Script are executed in Runtime in succession after being triggered. When an action is triggered while another action is in progress, the second action is kept in a queue until it can be executed.
Actions in Global Script and Graphics Designer do not affect each other.

Note
If the action should not be executed at each event, it is possible to define a condition in the action, the result of which controls whether the action is executed or not.
Notes on configuring triggers

Depending on the system, it cannot be guaranteed that an action with a cyclic trigger will be carried out at exactly the specified time. If this is a requirement, then the task (such as a check) should be implemented on the automation device.

The tag triggers should have priority over cyclic triggers: With cyclic actions, the action is always executed, e.g. every 20 seconds. The tag trigger only executes the action if a change in the value of the tag has been detected in the case of cyclic queries. This reduces the load on the system and increases performance.

If a tag trigger is used, configure the "Upon Change" cycle to start as seldom as possible. This query cycle causes the tag to trigger the action following every change. This causes high system loads.

Linking tag triggers

The CrossReference from WinCC can be used to quickly find all the application points of tags, even in VBS actions. Tag triggers in actions in Graphics Designer can be "linked" using CrossReference, i.e. replaced by other tags at all or selected points.

Use the standard formulations

```
HMIRuntime.BaseScreenName = "Screenname"
HMIRuntime.Tags("Tagname")
```

when addressing pictures and tags in your code in order to ensure that the pictures and tags are registered by the CrossReference.

Further information on CrossReference is available in the WinCC documentation.
1.10.7.2 Animation trigger

**Introduction**

As of WinCC V7.0, the "animation cycle" trigger art is available for the dynamization of objects with VBS. The animation cycle allows you to switch actions on and off in Runtime and to change the time in which the trigger is executed.

**Animation cycles**

<table>
<thead>
<tr>
<th>Name</th>
<th>Cycle</th>
<th>Name</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>CycleTime125ms</td>
<td>125 ms</td>
<td>CycleUser1</td>
<td>User cycle 1</td>
</tr>
<tr>
<td>CycleTime250ms</td>
<td>250 ms</td>
<td>CycleUser2</td>
<td>User cycle 2</td>
</tr>
<tr>
<td>CycleTime500ms</td>
<td>500 ms</td>
<td>CycleUser3</td>
<td>User cycle 3</td>
</tr>
<tr>
<td>CycleTime1s</td>
<td>1 s</td>
<td>CycleUser4</td>
<td>User cycle 4</td>
</tr>
<tr>
<td>CycleTime2s</td>
<td>2 s</td>
<td>CycleUser5</td>
<td>User cycle 5</td>
</tr>
<tr>
<td>CycleTime5s</td>
<td>5 s</td>
<td>CyclePicture</td>
<td>Picture cycle</td>
</tr>
<tr>
<td>CycleTime10s</td>
<td>10 s</td>
<td>CycleWindow</td>
<td>Window Cycle</td>
</tr>
<tr>
<td>CycleTime1min</td>
<td>1 min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CycleTime5min</td>
<td>5 min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CycleTime10min</td>
<td>10 min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CycleTime1h</td>
<td>1 h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You use the trigger by writing an action and using the "animation cycle" trigger type. This action can be activated or deactivated in Runtime with the "ActivateDynamic" and "DeactivateDynamic" methods. The methods are described in the VBS reference of the WinCC Information System. The correct syntax of the methods deviates from the description in the VBS reference and is shown in the following two examples.

**Example**

With an action at the determined property "Position X" (left), the rectangle is shifted 5 pixels to the right. Select the "animation cycle" event in the action as the trigger.
Enter the following as action in the "Left" property:

\[
\text{item.Left} = \text{item.left} + 5
\]

You can switch the action on and off at the property "Position X" with the following methods.

The trigger is switched on in Runtime with the "ActivateDynamic" method:

```vbs
Dim obj
Set obj = ScreenItems.Item("Rectangle1")
obj.ActivateDynamic "Left", "CycleTimels"
```

The trigger is switched off in Runtime with the "DeactivateDynamic" method:

```vbs
Dim obj
Set obj = ScreenItems.Item("Rectangle1")
obj.DeactivateDynamic "Left"
```

---

**Note**

The WinCC tags remain requested even when the trigger is switched off.

---

**See also**

ActivateDynamic method (Page 713)
1.10.7.3 How to add a trigger of the type "Timer"

Introduction

"Timer" type triggers execute an action at a certain time. "Timer" type triggers can be cyclic or acyclic triggers.

- Acyclic triggers Trigger an action once at the configured time.
- Cyclic triggers Trigger an action at regular intervals. The time interval and start time for the time must be triggered. If a standard cycle is selected as the cyclic trigger, the start time is always the start of Runtime. User-specific cycles can also be selected as standard cycles.

Note

Cyclic triggers guarantee a high updating rate of the system but require high system loads. Choose cyclic triggers only for those actions, where the update is very important. With high system loads, some actions may not be executable.

"Timer" type triggers are used to make the properties in Graphics Designer dynamic and execute global actions.

Procedure

1. Open the action.
2. Click the button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info". The "Properties..." dialog appears.
3. Select the "Triggers" tab.
4. Select the "Timer" trigger and then select the trigger type to be created: cyclic or acyclic.
5. Click on the "Add" button. The "Add Trigger" dialog appears.
6. If the "acyclic" trigger type has been selected: Enter a relevant trigger name and define the time at which the action should be executed.

7. If the "cyclic" trigger type has been selected: Enter a relevant trigger name and define the start time at which the action should be executed for the first time. Enter a cycle at which the action should be repeated.
   Click OK to confirm your entries.

---

**Note**

An action can be assigned several triggers. The action is always executed when one of the triggering events occurs.

---

**See also**

How to delete a trigger (Page 75)
How to add a trigger of the type "Tag" (Page 72)
Triggers (Page 64)
Creating and Editing Actions (Page 52)
Actions (Page 18)
1.10.7.4 How to add a trigger of the type "Tag"

**Introduction**

"Tag" type triggers execute an action following the change of a tag value. Any internal or external tag stored in WinCC can be used as a trigger tag.

Actions with tag triggers can be executed at the following times:

- **On change of tag:** The action is executed each time the tag value changes. Since this setting causes a very high system utilization, the updating rate should be set as low as possible.

- **Query the tag status according to standard cycle (including user cycles):** Define a cycle in whose intervals the tag value should be queried. The action is only executed when the tag value has changed when queried. When the query status is a large value, it is possible that the tag value changes but it is not detected by the system. In this case the action will not be performed.

If an action is linked with several tags, the action is executed when one of the tag values changes.

**Procedure**

1. Open the action.

2. Click the button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info". The "Properties..." dialog appears.

3. Select the "Triggers" tab.

4. Select "Trigger" as the tag.

5. Click on the "Add" button. The "Add Trigger" dialog appears.
6. Enter the name of the tag to be used as the trigger or click the button beside the "Tag Name" field in order to select a tag from the tag selection dialog.

7. Double click on the "Standard cycle" field to open the selection dialog for the tag update cycle:

Select a cycle and click on OK to confirm the selection.
1.10.7.5 How to change a trigger

Introduction

A defined trigger can be modified at any time, even during Runtime.

Procedure

1. Open the action whose triggers should be modified.

2. Click the button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info/Trigger". The "Properties..." dialog appears. Alternatively, call in the dialog without executing the action by double clicking on the trigger in the navigation window.

3. Select the "Triggers" tab.

4. Select the trigger to be modified and click the "Change" button.

5. Modify the trigger can confirm the entries with OK.
See also

- Triggers (Page 64)
- How to delete a trigger (Page 75)
- How to add a trigger of the type "Tag" (Page 72)
- How to add a trigger of the type "Timer" (Page 70)
- Creating and Editing Actions (Page 52)
- Actions (Page 18)

1.10.7.6 How to delete a trigger

Introduction

Defined triggers can be deleted at any time. Triggers can also be deleted during Runtime. If a trigger is deleted in Runtime, it only takes effect after the action is saved.

Note

Actions without trigger are not executed in Runtime. None of the actions which used the deleted trigger are executed any longer.

Procedure

1. Open the Global Script Editor or the Graphics Designer action editor.
2. Open the action.
3. Click the "Info/Trigger" toolbar button or select the "Info/Trigger" menu command. The "Properties..." dialog appears.
4. Select the "Triggers" tab.
5. Select the trigger to be deleted and click the "Delete" button.

6. The trigger is deleted immediately.

Note
Triggers can also be deleted directly in the Global Script navigation window using the "Delete" command in the context menu.

See also
Actions (Page 18)
How to change a trigger (Page 74)
How to add a trigger of the type "Tag" (Page 72)
How to add a trigger of the type "Timer" (Page 70)
Triggers (Page 64)
Creating and Editing Actions (Page 52)

1.10.8 How to Rename an Action

Introduction
Actions can be renamed in Global Script. When an action is renamed, the action name and file name are changed.
The action to be renamed must not be open in the editor window.

Procedure

1. Open Global Script.
2. Select the name of the action to be renamed in the editor's navigation window.
3. Select the "Rename" command from the context menu.
4. Enter a new name for the action with the extension *.bac.

See also

Protecting an Action with a Password (Page 62)
Saving Actions (Page 63)
How to add action-related information (Page 60)
How to Edit Actions (Page 57)
Creating a New Action (Page 56)
Triggers (Page 64)
Creating and Editing Actions (Page 52)
Actions (Page 18)
1.11 How to activate global actions in Runtime

Introduction

Scripts defined in Global Script are always executed when the configured trigger occurs. Scripts in the graphical Runtime system are executed when the picture is called in and the configured event or trigger occurs.

In order that the picture-independent, Global Script global actions can be executed, the Global Script Editor must be registered in the startup list of the Runtime computer.

Procedure

1. Select the "Properties" command in the computer's context menu WinCC Explorer. The "Computer properties" dialog will open.

2. Click on the "Startup" tab

3. Selection option "Global Script Runtime".

4. Click OK to confirm your entries.
See also

Creating and Editing Actions (Page 52)
Creating and Editing Procedures (Page 36)
VBScript Editors (Page 26)
Using Visual Basic Script in WinCC (Page 12)
1.12 Diagnostics

1.12.1 Diagnostics

Introduction
If the scripts are executed and tested in Runtime, the Diagnostics window can be used to display an analysis quickly.

Diagnostics Tools
WinCC provides a range of tools with which to analyze the behavior of actions in Runtime:

- The GSC Runtime and GSC Diagnostics application windows
- Use of a debugger

GSC Runtime and GSC Diagnostics
The GSC Runtime and GSC Diagnostics application window are used by inserting them in a process screen. This can be a process screen developed for diagnostics purposes which is called in Runtime.

The application windows are used for different strategies:
While Runtime is active, GSC Runtime provides information on the dynamic behavior of all (Global Script) actions, enables the individual startup as well as log on and off of each individual action and offers the access point to the Global Script Editor.

GSC Diagnostics issues the Trace methods contained in the actions in the chronological sequence they are called. This also applies to Trace instructions in procedures which are called in actions. The targeted implementation of Trace instructions, e.g. for the output of tag values, enables the progress of actions and the procedures called in them to be traced. The Trace instructions are entered in the form "HMI{runtime}.Trace(<Ausgabe>)".

The GSC Diagnostics displays trace output from C and VBS.

Note
Runtime errors in VBS are not displayed
Some script errors are neither output via trace nor displayed via the error dialog.

Debugger
You can use the "Microsoft Visual Studio 2008" debugger to test your VB scripts in Runtime.

The debugger is included in the WinCC Setup. If another Visual Studio version is already installed, Microsoft Visual Studio 2010 for example, use the debugger of this version.
See also

Testing with the Debugger (Page 87)
GSC Runtime (Page 84)
GSC Diagnostics (Page 81)

1.12.2 GSC Diagnostics

1.12.2.1 GSC Diagnostics

Introduction

GSC Diagnostics displays the chronological sequence of calls of the trace methods contained in the actions in the Diagnostics window. This also applies to Trace instructions in procedures which are called in actions. The targeted implementation of Trace instructions, e.g. for the output of tag values, enables the progress of actions and the procedures called in them to be traced.

Application

In order to use GSC Diagnostics, insert a GSC Diagnostics type application window in a process screen. The GSC Diagnostics attributes can be used to control the appearance of the GSC Diagnostics window.

In the case of a picture change, the content of the GSC Diagnostics window is deleted.

Note

Messages are also displayed in the "GSC Diagnostics" window when the debugger is activated.

See also

GSC Diagnostics Toolbar (Page 83)
GSC Diagnostics Attributes (Page 82)
Inserting the GSC Diagnostics Window into a Picture (Page 82)
1.12.2.2 Inserting the GSC Diagnostics Window into a Picture

Introduction

In order to use GSC Diagnostics, insert a GSC Diagnostics process screen. The process screen can be an existing picture or a picture which serves customized diagnostics purposes. GSC Diagnostics cannot be inserted directly in the process screen as an application but is inserted as an application in an application window. In this case, the application window is a component part of the process screen.

Requirements

Graphics Designer has been started and the process screen is open.

Procedure

1. Use the "Smart Objects" object palette to insert the "Application Window" in the picture.
2. Select the "Global Script" option from the "Window Contents" dialog and confirm the selection with "OK".
3. Select the "GSC Diagnostics" option from the "Templates" dialog.
4. Confirm the selection with OK in order to insert the Diagnostics window.

See also

GSC Diagnostics Toolbar (Page 83)
GSC Diagnostics Attributes (Page 82)
GSC Diagnostics (Page 81)

1.12.2.3 GSC Diagnostics Attributes

Overview

GSC Diagnostics has attributes which affect the appearance of the GSC Diagnostics window in Runtime. These relate to the geometric attributes, particularly to the following:

- Display: This attribute defines whether the window should be visible or hidden. The attribute can be made dynamic with the name Visible.
- Sizeable: This attribute defines whether the size of the window should be changeable in Runtime.
- Movable: This attribute defines whether the window should be moveable or not during Runtime.
- Border: This attribute defines whether the without is provided with a border. If the window has a border, its height and width can be modified in Runtime.
- Title: This defines whether the window has a title bar.
• Can be maximized: This attribute defines whether the title bar should contain the button to maximize the window.

• Can be closed: This attribute defines whether the title bar should contain the button to close the window.

• Foreground: This attribute defines whether the window should always be in the foreground.

See also

GSC Diagnostics Toolbar (Page 83)
Inserting the GSC Diagnostics Window into a Picture (Page 82)
GSC Diagnostics (Page 81)

1.12.2.4 GSC Diagnostics Toolbar

Overview

The GSC Diagnostics toolbar enables the output in the diagnostics window to be controlled and to save, print and open window content:

1: Delete the content of the diagnostics window
2: Stop the window being updated
3: Activate the window being updated
4: Opens a text file in the window
5: Saves the window contents in a text file
6: Print the window contents

See also

GSC Diagnostics Attributes (Page 82)
Inserting the GSC Diagnostics Window into a Picture (Page 82)
GSC Diagnostics (Page 81)
1.12.3 GSC Runtime

1.12.3.1 GSC Runtime

Introduction

GSC Runtime is a window which displays the dynamic behavior of all Global Script actions in Runtime. In addition, GSC Runtime can also be used during Runtime to influence the execution of each individual action and provide access to the Global Script editor.

Actions

C actions and VBS actions are differentiated in the GSC Runtime window:

- Symbolizes a C action
- Symbolizes a VBS action

The following information is issued:

- Action name: The name of the action
- ID: Action ID. They are used internally by the system. GSC Runtime supplies the corresponding action name together with the Action ID. The link between ID and action name is only valid until Runtime is stopped or, during Runtime, until an action is saved.
- Status: Provides information on the current status of the action. Refer to the table below for the possible statuses.
- Activation Interval: The time in the form Hour:Minute:Second, which should elapse between the action being called.
- Return Value: The return value of the action
- Started On: Date and time the current action was started
- Next Start: Date and time the action will be started again
- Error message: Contains the error text in the case of an error

Actions Status

Possible action status:

- Action was activated.
- Action was deactivated
- Action was stopped.
- Action in progress
- Error logging on the action!
- Error executing the action!
**Pop-Up Menu**

The following functions are available for every action in the pop-up menu:

- **Log off**: The relevant action will not be executed again when the current execution has finished.
- **Log on**: The relevant action will be executed again when the next trigger event occurs.
- **Start**: The relevant action will be executed once.
- **Edit**: The relevant action will be opened in the Global Script editor for editing. Runtime will remain active. If the edited action is compiled (when necessary) and saved the changes will be applied by the Runtime system immediately.

The option of opening the pop-up menu for every action can be controlled by assigning an authorization.

In order to use GSC Runtime, insert a GSC Runtime type application window in a process screen. The GSC Runtime attributes can be used to control the appearance of the GSC Runtime window.

---

**Note**

Updating the GSC Runtime window increases the system load. The system load is dependent on how many actions are visible in the window. The system load can be lowered by reducing the height of the window so that fewer lines are visible.

---

**See also**

- How to insert the GSC Runtime Window into a Picture (Page 85)
- GSC Runtime Attributes (Page 86)

---

**1.12.3.2 How to insert the GSC Runtime Window into a Picture**

**Introduction**

In order to use GSC Runtime, insert a GSC Runtime process screen. The process screen can be an existing picture or a picture which serves customized diagnostics purposes. GSC Runtime cannot be inserted directly in the process screen but is inserted as an application in an application window. In this case, the application window is a component part of the process screen.

**Requirements**

Graphics Designer has been started and the process screen is open.
Procedure

1. Use the "Smart Objects" object palette to insert the "Application Window" in the picture.
2. Select the "Global Script" option from the "Window Contents" dialog and confirm the selection with "OK".
3. Select the "GSC Runtime" option from the "Templates" dialog.
4. Confirm the selection with OK in order to insert the Diagnostics window.

See also

GSC Runtime (Page 84)
GSC Runtime Attributes (Page 86)

1.12.3.3 GSC Runtime Attributes

Overview

GSC Runtime has attributes which affect the appearance of the GSC Runtime window in Runtime. These relate to the geometric attributes, particularly to the following:

- **Display**: This attribute defines whether the window should be visible or hidden. The attribute can be made dynamic with the name Visible.
- **Sizeable**: This attribute defines whether the size of the window should be changeable in Runtime.
- **Movable**: This attribute defines whether the window should be moveable or not during Runtime.
- **Border**: This attribute defines whether the without is provided with a border. If the window has a border, its height and width can be modified in Runtime.
- **Title**: This defines whether the window has a title bar.
- **Can be maximized**: This attribute defines whether the title bar should contain the button to maximize the window.
- **Can be closed**: This attribute defines whether the title bar should contain the button to close the window.
- **Foreground**: This attribute defines whether the window should always be in the foreground.

See also

GSC Runtime (Page 84)
How to insert the GSC Runtime Window into a Picture (Page 85)
1.12.4 Testing with the Debugger

1.12.4.1 Testing with the Debugger

Overview

You can use the "Microsoft Visual Studio 2008" debugger to test your VB scripts in Runtime.

The debugger is included in the WinCC Setup. If another Visual Studio version is already installed, Microsoft Visual Studio 2010 for example, use the debugger of this version.

Installing the debugger for WinCC

To use the debugger in WinCC, you must start Setup and specify instance "Visual Studio 2008" as the default.

During the installation of WinCC, the "VS 2008 Shell Redist" folder is created on a drive of the PC, for example, as "D:\VS 2008 Shell Redist".

Procedure

1. Start the setup with the following file:
   – VS 2008 Shell Redist\Integrated Mode\Vside.enu.exe
2. Follow the instructions and accept the default settings.
   The following program is installed:
   – Microsoft Visual Studio 2008 Shell (Integrated Mode) - ENU
3. When you start the debugger the first time, the "Visual Studio Just-In-Time Debugger" dialog opens.
   Select the entry "New instance of Visual Studio 2008".
   To specify "Visual Studio 2008" as the default debugger, select the "Set the currently selected debugger as the default." setting.
Microsoft Visual Studio 2008 Debugger

See also

- Principles of Debugging (Page 90)
- Executing Script Commands (Page 102)
- How to Determine and Modify Tag and Property Values (Page 101)
- How to Set Bookmarks in Scripts (Page 100)
- Deleting Breakpoints (Page 99)
- Setting Breakpoints (Page 98)
- Processing Scripts Step-by-Step (Page 97)
- Selecting a Script for Editing (Page 96)
1.12.4.2 How to Activate the Debugger

Principle

There are several ways of activating the debugger:
- Automatic activation of the debugger when an error occurs in Runtime.
- Opening an error box in Runtime via which the debugger can be activated.
- Starting the debugger from the Start menu and opening a running Runtime scripts.

Procedure

The following procedure describes the first two points, activating the debugger in WinCC.

1. In the computer's pop-up menu in WinCC Explorer, select the command "Properties". The "Computer Properties" dialog appears.
2. Select the "Runtime" tab control.
3. Activate the required debug options. The debug behavior for actions in Global Script and Graphics Designer can be set independently of each other:
4. Select "Start debugger" when the debugger should be started directly following an error in the Runtime.
5. Select "Display Error Dialog", if you do not want to start the Debugger directly but wish to display an error dialog with information about the error. The debugger can be started from the error box by means of a button.
6. Click OK to confirm your entries.

Starting the Debugger and Opening a Running Script

The debugger can also be started up later and linked to the system currently running. Define a connection in the debugger to the respective processes, "pdlrt.exe" for the Graphical Runtime System and "gscrt.exe" for the Global Runtime System. The way to open a running script in the debugger is described under the topic "Select Script".
Exiting the Debuggers

It is possible to stop the debugger without exiting the WinCC Runtime.

See also

- How to Set Bookmarks in Scripts (Page 100)
- Executing Script Commands (Page 102)
- How to Determine and Modify Tag and Property Values (Page 101)
- Deleting Breakpoints (Page 99)
- Setting Breakpoints (Page 98)
- Processing Scripts Step-by-Step (Page 97)
- Selecting a Script for Editing (Page 96)
- Action and Procedure Names in the Debugger (Page 94)
- Structure of VBScript Files (Page 92)
- Principles of Debugging (Page 90)
- Testing with the Debugger (Page 87)
- Diagnostics (Page 80)

1.12.4.3 Principles of Debugging

Introduction

You can use the “Microsoft Visual Studio 2008” debugger to test your VB scripts in Runtime. The debugger is included in the WinCC Setup. If another Visual Studio version is already installed, Microsoft Visual Studio 2010 for example, use the debugger of this version.

The debugger can be used to:

- View the script source code to be debugged
- Step-by-step processing of the scripts to be checked
- Display and modify tag and property values
- View and monitor the script progress

**Note**

Please note that the code displayed in the debugger is write-protected. The code cannot be changed directly in the debugger but only test the necessary changes.

Error types

A distinction is made between the following types of error by the debug:
Syntax errors

Syntax errors occur, for example, when a key word is written incorrectly or a parenthesis is not closed. When a syntax check from WinCC is used, syntax errors can be excluded before testing the scripts in Runtime. In principle, only syntactically correct scripts can be saved in Graphics Designer. The WinCC syntax check also checks:

- Whether the procedure names are unique in Global Script
- Whether an action module in Global Script contains only one procedure
- Whether the action part in Graphics Designer contains only one procedure

As a result of the syntax check in WinCC, the script is parsed without being executed. The script is parsed again directly before executing in Runtime. All the script parts are parsed, even those which are executed after a certain action has been executed at a later time.

If the script contains syntax errors, the script is not executed in Runtime.

Runtime error

A Runtime error occurs when an attempt is made to execute an invalid/erroneous action, e.g. because a tag has not been defined. In order to intercept Runtime errors, use the "On Error Resume Next" command in the VBScript. The command causes the subsequent command to be executed following a Runtime error. The error code can subsequently be checked using the Err object. In order to deactivate the processing of Runtime errors in the script, use the "On Error Goto 0" command.

Logical errors

The debugger is particularly helpful in clearing up logical errors. A logical error occurs when an unexpected result is received because, for example, a condition was incorrectly checked. To clear logical errors, go through the scripts step-by-step in order to detect the part which does not function properly.

Basic Procedure

When an error has occurred and the debugger is open, the script appears in a window, write-protected. It is possible to navigate through the script document, set breakpoints, execute the script again in Runtime and to process the script step-by-step.

The most important steps for successful debugging of the scripts are described under "Processing Scripts Step-by-Step".

The source codes of the scripts cannot be edited directly in the scripts. When an error has been detected, the error can be corrected in the original script in WinCC, e.g. load the picture again and update it in the debugger.

Note

Tips and tricks for debugging, frequently occurring error codes and other information is available in the Microsoft Script Debugger online help.
Change Picture During Debug

If a picture change is executed during debugging, the script document of the "old" picture remains open but is no longer valid. If necessary, invalid errors are displayed because the objects called following the picture change are no longer available.

See also

Testing with the Debugger (Page 87)
Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Processing Scripts Step-by-Step (Page 97)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
How to Activate the Debugger (Page 89)
Diagnostics (Page 80)

1.12.4.4 Structure of VBScript Files

Principle

In order not to hinder the simultaneous processing of cyclic and event-driven scripts in the graphical Runtime system, the event-driven actions and cyclic/tag-driven actions are strictly separated during processing. In this way, a cyclic action, for example, cannot hinder the execution of an action initiated by clicking a button.

To ensure this, the event-driven actions and the cyclic/tag-driven actions are stored in separate script files when saving a picture. If a global picture section has been defined in actions in Graphics Designer, this is copied into both scripts. In the same way, modules which are used in an action are also copied in both script files.

If a tag from a module should be used, the corresponding module must be called in. Otherwise, the module is not copied in the script file and an error is generated.

Note

Since the two script files are handled separately, they have no common data area. Therefore, there is no synchronization of global tags between the two script files. If synchronization is required, implement this using the DataSet object or internal WinCC tags.
Structure of the Script Files

When scripts are debugged with a debugger, the script files always open the different Runtime systems.

In the case of the graphical Runtime system, this means that you receive two script files per picture:

- `<Bildname>.pdl_events`: Contains the event-driven actions.
- `<Bildname>.pdl_triggers`: Contains the cyclic and tag-controlled actions.

The following section describes how the script files are structured:

**Graphical Runtime system**

<table>
<thead>
<tr>
<th>Script file contents &lt;picture name&gt;.pdl_events</th>
<th>Script file contents &lt;picture name&gt;.pdl_triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>General code</td>
<td>General code</td>
</tr>
<tr>
<td>Standard modules (if used)</td>
<td>Standard modules (if used)</td>
</tr>
<tr>
<td>Project modules (if used)</td>
<td>Project modules (if used)</td>
</tr>
<tr>
<td>Event-controlled actions</td>
<td>Cyclical or tag-triggered actions</td>
</tr>
</tbody>
</table>

**Global Script Runtime**

<table>
<thead>
<tr>
<th>Global Script Runtime environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required standard module 1</td>
</tr>
<tr>
<td>Required standard module n</td>
</tr>
<tr>
<td>Required project module 1</td>
</tr>
<tr>
<td>Required project module n</td>
</tr>
<tr>
<td>Action 1</td>
</tr>
<tr>
<td>Action n</td>
</tr>
</tbody>
</table>
Note
Please note that the actions and procedures of the graphic Runtime system are not displayed with the action name in the script file under which it was saved in WinCC. The name conventions for actions and procedures in the script files state described in "Action and Procedure Names in the Debugger".

See also
How to Activate the Debugger (Page 89)
Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Processing Scripts Step-by-Step (Page 97)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Principles of Debugging (Page 90)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)

1.12.4.5 Action and Procedure Names in the Debugger

Action and Procedure Names in the Debugger

The names of procedures and actions in debugger script files differ from the names under which they were saved by the scripts in WinCC.

The action and procedure names in the script files are compiled according to the following rules:

<table>
<thead>
<tr>
<th>Action type</th>
<th>Name of the script file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclic or tag-driven actions on a property</td>
<td>ObjectName_PropertyName_Trigger</td>
</tr>
<tr>
<td>Mouse events</td>
<td>ObjectName_OnClick</td>
</tr>
<tr>
<td></td>
<td>ObjectName_OnLButtonDown</td>
</tr>
<tr>
<td></td>
<td>ObjectName_OnLButtonUp</td>
</tr>
<tr>
<td></td>
<td>ObjectName_OnRButtonDown</td>
</tr>
<tr>
<td></td>
<td>ObjectName_OnRButtonUp</td>
</tr>
<tr>
<td>Action type</td>
<td>Name of the script file</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>Keyboard events</td>
<td>ObjektName_OnKeyDown</td>
</tr>
<tr>
<td></td>
<td>ObjektName_OnKeyUp</td>
</tr>
<tr>
<td>Object events</td>
<td>ObjektName_OnObjectChanged</td>
</tr>
<tr>
<td></td>
<td>ObjektName_OnSetFocus</td>
</tr>
<tr>
<td>Events on properties</td>
<td>ObjektName_PropertyName_OnPropertyChanged</td>
</tr>
<tr>
<td></td>
<td>ObjektName_PropertyName_OnPropertyStateChanged</td>
</tr>
<tr>
<td>Picture events</td>
<td>Document_OnOpen</td>
</tr>
<tr>
<td></td>
<td>Document_OnClosed</td>
</tr>
</tbody>
</table>

**Permitted length of action names**

The names of the actions in the script files are limited to 255 characters. Each special character used in an object name is converted to five characters. The special characters are represented by a four-place hexadecimal code behind the preceding X. If, for example, an action is configured on a button with the name "PushHere" per mouse click, the script in the script file appears as "PushHere_OnClick".

If the object name compiled is too long, an error message is issued during the syntax check in WinCC. As a result of this restriction, graphic object names cannot be selected with any length during configuration.

**Note**

If you wish to determine the name of an object in Runtime, press <CTRL+ALT+SHIFT> and position the mouse over the corresponding object. The picture name and object name then appears in a tooltip.

**See also**

- Executing Script Commands (Page 102)
- How to Determine and Modify Tag and Property Values (Page 101)
- How to Set Bookmarks in Scripts (Page 100)
- Deleting Breakpoints (Page 99)
- Setting Breakpoints (Page 98)
- Processing Scripts Step-by-Step (Page 97)
- Selecting a Script for Editing (Page 96)
- Structure of VBScript Files (Page 92)
- Principles of Debugging (Page 90)
- How to Activate the Debugger (Page 89)
1.12.4.6 Selecting a Script for Editing

Introduction

If the Microsoft Script Debugger is called in Windows instead of automatic activation using WinCC, scripts which are currently running can be called in for editing in Runtime.

Requirement

Runtime is activated, the picture to be debugged is active.

Procedure

2. Create a new script file:
   – File > New > File… > Select script template "Windows Script Host".
3. Start debugging with "Debug > Attach to Process".
   Select, for example, gscrt.exe or pdlrt.exe.

See also

Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Processing Scripts Step-by-Step (Page 97)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
Principles of Debugging (Page 90)
How to Activate the Debugger (Page 89)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)
1.12.4.7 Processing Scripts Step-by-Step

Introduction

The Microsoft Script Debugger can be used to process the scripts step-by-step in order, for example, to locate logical errors systematically. The effect of each individual script line can be tested in Runtime.

The Procedure in Principle

1. Activate the document to be debugged in Runtime.
2. Start the debugger manually from the Start menu and open the required script file or activate the debugger in WinCC. When activated in WinCC, the debugger automatically opens when an attempt is made to execute an erroneous script.
3. Set a breakpoint in the script file. Breakpoints are normally set in front of code lines in which errors are suspected.
4. Switch to WinCC Runtime and trigger an action which causes the script to run. The Debugger stops at the first breakpoint and marks the current line.
5. In order to go through the script document step-by-step, select one of the following menu commands:
   "Debug" > "Step Into": Skip to the next code line. If the script calls a procedure in this line, it skips to the procedure using the "Step Into" command. The procedure called can then be processed step-by-step.
   "Debug" > "Step Over": Skips the procedure called. The procedure is called but the debugger does not stop at the individual lines of the procedure. Instead, it moves to the next line of the current script after the procedure has been executed.
6. To interrupt the step-by-step processing of a procedure, select the "Debug" > "Step Out" menu commands. The debugger then skips to the next action.
7. Proceed step-by-step to the end of the document or select the "Debug" > "Run" menu items to start the script again in Runtime.

See also

Principles of Debugging (Page 90)
Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
1.12.4.8 Setting Breakpoints

Introduction

Breakpoints can be set in a script to stop at specific points when processing it and to start the debugger. Set a breakpoint in front of a line, for example, which you suspect contains a script error.

It is possible to:

- Set breakpoints at specific lines to locate logical errors in the script step-by-step.
- Set a breakpoint and call the debugger before the next line in the script is processed. These procedure is used, for example, for events such as "Change picture".

When a script file is updated in the debugger, all the breakpoints are lost.

If a breakpoint is set in one of the script files "<Bildname>.pdl_trigger" or "<Bildname>.pdl_event", all the trigger-driven or all event-driven procedures are stopped, respectively, in Runtime.

Requirements

Runtime is activated, the picture to be debugged is active.

Procedure

Setting a breakpoint

1. Start the debugger and select the script. If automatic activation of the debuggers in WinCC has been selected, the debugger is called in as soon as an erroneous script is executed.

2. Position the cursor on the action in which a breakpoint should be set.

3. Open the "Debug" menu and select the "Toggle Breakpoint" item or the icon from the toolbar.
   The next executable line will be marked by a red dot.

4. Switch to WinCC Runtime and execute the action you wish to debug.
   The Debugger stops at the first breakpoint it finds in the script. The current line is displayed on a yellow background. The script can then be processed step-by-step.

See also

Deleting Breakpoints (Page 99)
Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
1.12.4.9 Deleting Breakpoints

Introduction

When an error has been cleared properly, the breakpoints in a script can be cleared individually or all together.

Procedure

1. Position the cursor in the line whose breakpoint is to be deleted.
2. Open the "Debug" menu and select the "Toggle Breakpoint" item or the icon from the toolbar.
The next line will be displayed without a mark.
3. To delete all the breakpoints in a script, open the "Debug" menu and select the "Clear all Breakpoints" entry or the icon from the toolbar.

See also

Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Setting Breakpoints (Page 98)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
Principles of Debugging (Page 90)
How to Activate the Debugger (Page 89)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)
1.12.4.10 How to Set Bookmarks in Scripts

Introduction

During the debug routine, bookmarks can be set on code lines so that they can be found easier again one line later.

Setting or deleting bookmarks

Position the mouse pointer in the row where you wish to set a bookmark, and click <CTRL+F9> to set a bookmark or <CTRL+SHIFT+F9> to delete one.

Skipping to the next bookmark

Press <F9> to go to the next bookmark in the script.

Skipping to the previous bookmark

Press <SHIFT+F9> to go to the previous bookmark in the script.

See also

Executing Script Commands (Page 102)
How to Determine and Modify Tag and Property Values (Page 101)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
Principles of Debugging (Page 90)
How to Activate the Debugger (Page 89)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)
1.12.4.11 How to Determine and Modify Tag and Property Values

Introduction

While a script is running in Runtime, the "Command Window" of the debugger can be used, for example, to compile and modify values of tags or properties in the script currently running. It is possible, for example, to reset a process value for a script to zero without having to stop the process.

Note

If you wish to determine the name of a WinCC object in Runtime, click <CTRL+ALT+SHIFT> and position the mouse over the corresponding object. The picture name and object name then appears in a tooltip.

Requirements

The script runs in Runtime and the debugger is opened.

Procedure

1. Set at least one breakpoint in the current script.
2. Switch to WinCC Runtime and trigger an action which causes the script to be executed. The Debugger stops at the first breakpoint.
3. Open the "View" menu and activate the "Command Window" entry. The "Command Window" opens.
4. In order to determine the value of a tag or property, enter a "?" followed by a Space and the name of the tag or property whose value is to be determined, e.g. "?myTag". Press <RETURN> to execute the command.
5. In order to modify the value of a tag/property, assign a value in the VBS syntax.

See also

Principles of Debugging (Page 90)
Executing Script Commands (Page 102)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
How to Activate the Debugger (Page 89)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)

1.12.4.12 Executing Script Commands

Introduction

While a script is running in Runtime, the "Command Window" of the debugger can be used to execute script commands directly and thus manipulate the running of the current script. The script commands can be executed directly for test purposes without creating the command in a script and activating it. It is possible, for example:

- To retrieve methods
- To retrieve procedures
- To manipulate object properties

"Command Window" can basically be used to execute all commands which can also be executed from a VBScript.

Requirements

The script runs in Runtime and the debugger is opened.

Procedure

1. Set at least one breakpoint in the current script.
2. Switch to WinCC Runtime and trigger an action which causes the script to be executed. The Debugger stops at the first breakpoint.
3. Open the "View" menu and activate the "Command Window" entry. The "Command Window" opens.
4. Enter the required command and press "ENTER".

Note

If a faulty command is entered in the Command window, no error message is issued in Runtime. The message "<Script Error>" appears in the Command window instead.

See also

How to Determine and Modify Tag and Property Values (Page 101)
How to Set Bookmarks in Scripts (Page 100)
Deleting Breakpoints (Page 99)
Setting Breakpoints (Page 98)
Selecting a Script for Editing (Page 96)
Action and Procedure Names in the Debugger (Page 94)
Structure of VBScript Files (Page 92)
Principles of Debugging (Page 90)
How to Activate the Debugger (Page 89)
Testing with the Debugger (Page 87)
Diagnostics (Page 80)
1.13 Printing VBScripts

Principle

The actions and procedures configured in both Global Script and in Graphics Designer can be documented in WinCC.

The documentation options are distinguished between:

- **Print Feedback Doc:** In Graphics Designer, all the configured actions are printed with the feedback of the current picture. The Feed Back contains the C-actions and VBS actions, located beside each other, differentiated by the source text (C or VBScript).

- **Print current script:** The Feed Back in Global Script always contains the currently open procedure or action.

WinCC provided predefined print layouts for the layout of the Feed Back. Customized print layouts can also be developed and linked to the Print Job tab control with "Project Documentation - Setup".

Procedure

1. Open Global Script or Graphics Designer according to the scripts to be documented.
2. Configure the print job, if necessary, using the "Project Documentation - Setup" command.
3. Use the "View Project Documentation" command to preview the data to be printed.
4. Select the menu commands "File" > "Print Project Documentation" to print the data.

See also

- Creating and Editing Actions (Page 52)
- Creating and Editing Procedures (Page 36)
- VBScript Editors (Page 26)
- Using Visual Basic Script in WinCC (Page 12)
VBS object model in WinCC

The WinCC object model of the graphic Runtime system enables access to graphic objects and tags in Runtime.

When you click on an object name, you are shown a detailed description.
The VBS object model in a faceplate type

The VBS object model is not valid for WinCC in a Faceplate type. It is replaced by a completely new model.
The VBS object model of the Faceplate type provides you with access to the graphic objects and Faceplate tags of the Faceplate type in Runtime.

**Objects**

Objects and lists are provided for access to all the objects in the graphic Runtime systems: Graphic objects, pictures, layers and tags.

**Properties**

The properties of the individual objects can be used to modify specific graphic objects and tags in Runtime, e.g. activating an operating element per mouse click or triggering a color change by modifying a tag value.

**Methods**

Methods, which are applied to individual objects, can be used to read tag values for further processing or display diagnostics messages in Runtime.

**See also**

Object types of the ScreenItem object (Page 143)

Methods (Page 710)

Properties (Page 301)

Objects and Lists (Page 107)

### 1.14.2 Objects and Lists

### 1.14.2.1 Objects and Lists

**Overview**

The objects and lists provided in WinCC object models enables access to graphic objects and tags in Runtime.

**Navigation in Object Models**

Access is made to objects in the VBS object model in hierarchical sequence. If, for example, a picture element is accessed within a picture, access is made to the picture element in the picture via its parent object (the surrounding picture).
Example:

Access to Graphic Objects

In WinCC, access is made to pictures, layers and graphic objects in Runtime using the superordinate "HMIRuntime" object. Access to objects and layers is always made via the picture (screen) in which they are contained.

Access to Tags

In WinCC, tags are accessed directly in Runtime using the superordinate "HMIRuntime" object. Tag values can be read out or set anew.
Lists

Lists of WinCC object models behave in the same way as standard collections of VBS. Exception: The "Tags" list has no Enum function.

Available Objects

- Alarm
- Alarms
- AlarmLogs
- DataItem
- DataLogs
- DataSet
- HMIRuntime
- Item
- Layer
- Layers
- Logging
- ProcessValues
- ProcessValue
- Project
- ScreenItem
- ScreenItems
- Screen
- Screens
- Tag
- Tags
- TagSet

See also

ScreenItems Object (List) (Page 129)
TagSet Object (List) (Page 141)
Tags Object (List) (Page 140)
Tag Object (Page 137)
Screens Object (List) (Page 134)
Screen Object (Page 131)
ScreenItem Object (Page 125)
1.14.2.2 Alarm object

Description

The alarm object is used to access the Alarms object list.

Note

The properties of the alarm object are not automatically updated when the values of the properties change.

See also

Alarms object (list) (Page 110)

1.14.2.3 Alarms object (list)

Description

Use the alarm object to trigger existing messages.
Usage

Using the "Alarms" list you can:

- Access a message in the list (Item method)
- Create a new alarm object (Create method)
- Read the alarm ID of the message (AlarmID attribute)
- Set the status of a message (State property)
- Set the time stamp of the message (Timestamp property)
- Generate an instance of the alarm object (Instance property)
- Read the name of the computer on which the message came (ComputerName property)
- Read the name of the user who triggered the message (UserName property)
- Set the names of the process value blocks (ProcessValues property)
- Set the comment of the message (Comment property)
- Set the server prefix of the message (Context property)

Example

In the following example, the message with the alarm number "1" configured in the Alarm Logging Editor will be triggered:

```
'VBS360
Dim MyAlarm
Set MyAlarm = HMIRuntime.Alarms(1)
MyAlarm.State = 5 'hmiAlarmStateCome + hmiAlarmStateComment
MyAlarm.Comment = "MyComment"
MyAlarm.UserName = "Hans-Peter"
MyAlarm.ProcessValues(1) = "Process Value 1"
MyAlarm.ProcessValues(4) = "Process Value 4"
MyAlarm.Create "MyApplication"
```

See also

- TimeStamp Property (Page 622)
- ComputerName property (Page 391)
- Context property (Page 391)
- State property (Page 587)
- AlarmID property (Page 308)
- Instance property (Page 454)
- Comment property (Page 390)
- UserName property (Page 676)
Using the object, swapped archive segments of Alarm Logging may be reconnected to Runtime, or previously swapped archive segments of Alarm Logging may be deleted again. Therein

- Archive segments to be swapped are copied to the common archiving directory of the WinCC project, or
- previously swapped archive segments are deleted in the common archiving directory.

Using parameters you may control from where archive segments are to be swapped. You may also specify the time period over which archive segments are to be swapped or deleted. Archive segments are copied to the common archiving directory of the project.

If an error occurred during the operation with archiving segments, the method used returns an error message. Additional information may be found under the subject heading "Error Messages from Database Area".

**Usage**

Previously swapped archive segments of Alarm Logging may be connected with Runtime ("Restore" method).

Previously swapped archive segments of Alarm Logging may be deleted from the Runtime project ("Remove" method).

**Example:**

In the following example, archive segments from Alarm Logging are swapped and the return value is output as Trace.

```vbscript
'VBS187
```
See also

Error Messages from Database Area (Page 820)
Restore Method (Page 794)
Remove Method (Page 789)
DataLogs Object (Page 114)
Logging Object (Page 122)

1.14.2.5 DataItem Object

Description

The DataItem object is used to access the contents of the DataSet list. Values or object references are stored in the list as DataItem.

Access uses the name under which the value was added to the list. Single access using an index is not recommended since the index changes during adding or deleting of values. The index may be used to output the complete contents of the list. The output is in alphabetical order.

Note

For object references it must be ascertained that objects are multiread-enabled.

Example:

The example shows how the value of 'Motor1' is output as Trace.

'VBS163
HMI_Runtime.Trace "motor1: " & HMI_Runtime.DataSet("motor1").Value & vbNewLine
The following example enumerates all DataItem objects of the DataSet list. Name and value are output as Trace.

'VBS164
Dim data
For Each data In HMIRuntime.DataSet
HMIRuntime.Trace data.Name & ": " & data.Value & vbCrLf
Next

---

**Note**

For objects, value may possibly not be output directly

---

**See also**

Screen Object (Page 131)
HMIRuntime Object (Page 118)
DataSet Object (List) (Page 116)
Value Property (Page 680)
Name Property (Page 509)

---

### 1.14.2.6 DataLogs Object

**Description**

Using the object, swapped archive segments of Tag Logging may be reconnected to Runtime, or previously swapped archive segments of Tag Logging may be deleted again. Therein

- Archive segments to be swapped are copied to the common archiving directory of the WinCC project, or
- previously swapped archive segments are deleted in the common archiving directory.

Using parameters you may control from where archive segments are to be swapped. You may also specify the time period over which archive segments are to be swapped or deleted. In addition, you may set the archive type ("Tag Logging Fast", "Tag Logging Slow", "Tag Logging..."
Fast and Tag Logging Slow). Archive segments are copied to the common archiving directory of the project.

If an error occurred during the operation with archiving segments, the method used returns an error message. Additional information may be found under the subject heading "Error Messages from Database Area".

Usage

Previously swapped archive segments of Tag Logging may be connected with Runtime ("Restore" method).

Previously swapped archive segments of Tag Logging may be deleted from the Runtime project ("Remove" method).

Example:

In the following example, fast archive segments from Tag Logging are swapped and the return value is output as Trace.

'VB$188

See also

Error Messages from Database Area (Page 820)
Restore Method (Page 794)
Remove Method (Page 789)
AlarmLogs Object (Page 112)
Logging Object (Page 122)
1.14.2.7 DataSet Object (List)

Description

Using the DataSet object, data may be exchanged across several actions. A DataSet object is global and defined by the Screen object. Any VBS action may access the data. The DataSet object at the Screen object must be addressed according to picture hierarchy and shall persist as long as the picture is displayed. The global object persists over the entire Runtime time period. Access uses the DataItem object.

Note

Objects of type Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet cannot be included in the DataSet list. The DataSet object does not support any classes.

Usage

Using the "DataSet" list, you may:

- Output or process (enumerate) all objects in the list.
- Output the number of elements contained ("Count" property).
- To process a specific object in the list ("Item" method).
- Add an object to the list ("Add" method).
- Remove a specific object from the list ("Remove" method).
- Remove all objects from the list ("RemoveAll" method).

Access to list elements uses:

```
HMIRuntime.DataSet("Itemname")
```
For a picture-specific list, access uses:

HMIRuntime.Screens("Screenname").DataSet("Itemname")

In a picture, you may access the DataSet object of the picture by using:

DataSet("Itemname")

If upon access the stated name does not exist in the list, VT_Empty is returned and an Exception is triggered.

**Example:**

The example shows how to add a value to the list, how to read it and remove it. It make sense to perform this in several different actions.

'VBS162
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")

**See also**

DataItem Object (Page 113)
RemoveAll Method (Page 792)
Remove Method (Page 789)
Item Method (Page 770)
Count Property (Page 393)
Add Method (Page 713)
The HMIRuntime object represents the graphic Runtime environment.

Usage

The "HMIRuntime" object can be used for the following, for example:

- Read or set the current Runtime language ("Language" property).
- Read or set the name of the current base picture ("BaseScreenName" property).
- Read the path of the active Runtime project ("ActiveProject" property).
- Access tags ("Tags" property).
- Access tags of a list ("DataSet" property).
- Exit Runtime ("Stop" method).
- Display messages in a diagnostics window ("Trace" method).
Example:

The following command terminates WinCC Runtime:

'VBS3
HMIRuntime.Stop

See also

Screens Object (List) (Page 134)
TagSet Object (List) (Page 141)
Tags Object (List) (Page 140)
Logging Object (Page 122)
DataSet Object (List) (Page 116)
Visible Property (Page 696)
Trace Method (Page 811)
Tags Property (Page 597)
Stop Method (Page 810)
AlignmentLeft Property (Page 309)
Logging Property (Page 484)
Language Property (Page 458)
DataSet Property (Page 398)
CurrentContext Property (Page 394)
BaseScreenName Property (Page 342)
ActiveProject Property (Page 303)
ActiveScreen Property (Page 304)
MenuToolBarConfig Property (Page 495)
Alarms object (list) (Page 110)

1.14.2.9 Item Object

Description

The "Item" object provides a reference to the current object.
Usage

The "Item" object is used, for example, to address the properties of the object currently selected in Graphics Designer.

Example:

In the following example, a rectangle has been created. When the object has been selected, all the properties of the current object can be set a background color red:

'VBS195
Item.BackColor = RGB(255,0,0)

See also

Objects and Lists (Page 107)

1.14.2.10 Layer Object

Description

The layer object returns the result of access to the layers list.

Parent Object

Picture, in which the picture layer is.

Usage

Depending on certain events, the Layer object can be used to obtain access to the properties of a complete layer in order, for example, to hide or unhide a layer with operating elements according to the operator authorization.
The "Layer" object can be used to:

- To activate or deactivate the visualization of a layer ("Visible" property).
- To read out the name of a layer ("Name" property).

**Note**

The layer property specifies the layer in which the object is located. The layer "0" is output as "Layer0".

When accessed, the layers are counted up from 1 in VBS. Therefore, the layer "1" must be addressed with "layers(2)".

**Example:**

In the following example, Layer 1 is set invisible:

```vbs
'VBS4
Layers(2).Visible = vbFalse
```

**See also**

Layer Object (Page 120)
Visible Property (Page 696)
Parent Property (Page 529)
Name Property (Page 509)

1.14.2.11 Layers Object (Listing)

**Description**

The Layers list enables access to all 32 layers of the graphical Runtime system.

**Parent Object**

Picture, in which the picture layer is.
Usage

The "Layers" list can be used to:

- Process all layers in the list ("_NewEnum" property).
- Count all layers contained in the list ("Count" property).
- Process a layer from the list ("Item" method).

The properties represent default properties and methods of a list and are not described in detail in the WinCC documentation.

See also

Parent Property (Page 529)
Item Method (Page 770)
Count Property (Page 393)
Layer Object (Page 120)

1.14.2.12 Logging Object

Description

Using the object, swapped archive segments may be reconnected to Runtime, or previously swapped archive segments may be deleted again. Therein

- Archive segments to be swapped are copied to the common archiving directory of the WinCC project, or
- previously swapped archive segments are deleted in the common archiving directory.

Using parameters you may control from where archive segments are to be swapped. You may also specify the time period over which archive segments are to be swapped or deleted. Archive segments are copied to the common archiving directory of the project.

If an error occurred during the operation with archiving segments, the method used returns an error message. Additional information may be found under the subject heading "Error Messages from Database Area".

Usage

Previously swapped archive segments of Alarm Logging and Tag Logging may be connected with Runtime ("Restore" method).
Previously swapped archive segments of Alarm Logging and Tag Logging may be deleted from the Runtime project ("Remove" method).

**Example:**

In the following example, archive segments from Alarm Logging and Tag Logging are swapped and the return value is output as Trace.

```
'VBS189
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Folder","2004-09-14","2004-09-20",-1) & vbCrLf
```

**See also**

Error Messages from Database Area (Page 820)
DataLogs Object (Page 114)
AlarmLogs Object (Page 112)
Restore Method (Page 794)
Remove Method (Page 789)
DataLogs Property (Page 398)
AlarmLogs Property (Page 308)

### 1.14.2.13 ProcessValue Object

**Description**

The ProcessValue object is used to access the ProcessValues object list.

**Note**

Only the 10 predefined ProcessValues are supported.

**See also**

ProcessValues Object (List) (Page 124)
### 1.14.2.14 ProcessValues Object (List)

**Description**

<table>
<thead>
<tr>
<th>Alarms</th>
<th>ProcessValues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>ProcessValue</td>
</tr>
</tbody>
</table>

**Usage**

Using the "ProcessValues" list, you can:

- Edit a ProcessValue from the list ("Item" method)
- Display or edit all the objects in the list (_NewEnum attribute)
- Count all ProcessValues contained in the list (Count property)
- Read or set the values of the ProcessValue object (Value property)

The properties represent default properties and methods of a list and are not described in detail in the WinCC documentation.

**See also**

- Alarms object (list) (Page 110)
- ProcessValue Object (Page 123)
- Count Property (Page 393)
- Value Property (Page 680)
- Item Method (Page 770)

### 1.14.2.15 Project Object

**Description**

```
   HMIRuntime
    └── ActiveProject
```

Using the object, information may be requested from the current Runtime project.

The project object is returned as the result of ActiveProject.
Usage

Using the "Project" object, you may:

- Read the path of the current Runtime project ("Path" property).
- Read the name of the current Runtime project, without path or file extension ("Name" property).

Example:

The following example returns name and path of the current Runtime project as Trace:

```
'VBS159
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

See also

ActiveProject Property (Page 303)
Name Property (Page 509)
Path Property (Page 531)

1.14.2.16 ScreenItem Object

Description

The ScreenItem object returns the result of access to the ScreenItem list.

Parent Object

Picture containing the picture element.

Usage

The ScreenItem object can be used to access the properties of graphic objects within a picture according to certain events.
The "ScreenItem" object can be used for the following, for example:

- To activate or deactivate the visualization of an object ("Visible" property).
- To release or block the operation of an object ("Enabled" property).
- Change the width and height of an object ("Height" and "Width" properties).
- Change the position of an object ("Top" and "Left" properties).
- Read and define a layer in which a graphic object is located ("Layer" property).
- Read or define the name of a graphic object ("ObjectName" property).
- Define a reference to the superordinate picture ("Parent" property).

Using the "Activate" method, the focus is set on the respective ScreenItem object. If the focus cannot be set because the object is non-operable, for example, an error is generated. Using error processing (On Error Resume Next), the error may be evaluated.

### Possible features of ScreenItem

The "ScreenItem" object can contain the following object types:

<table>
<thead>
<tr>
<th>Standard objects</th>
<th>Smart objects</th>
<th>Windows objects</th>
<th>Tube objects</th>
<th>Controls</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ellipse</td>
<td>3D bar</td>
<td>Button</td>
<td>Double T-piece</td>
<td>Siemens HMI Symbol Library</td>
<td>Customized Object</td>
</tr>
<tr>
<td>Ellipse arc</td>
<td>Application window</td>
<td>Check box</td>
<td>Polygon tube</td>
<td>WinCC AlarmControl</td>
<td>Group</td>
</tr>
<tr>
<td>Ellipse segment</td>
<td>Bar</td>
<td>Radio box</td>
<td>Tube bend</td>
<td>WinCC digital/analog clock control</td>
<td></td>
</tr>
<tr>
<td>Circle</td>
<td>Picture window</td>
<td>Round button</td>
<td>T-piece</td>
<td>WinCC Function-TrendControl</td>
<td></td>
</tr>
<tr>
<td>Circular arc</td>
<td>Control</td>
<td>Slider</td>
<td>WinCC gauge control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pie segment</td>
<td>I/O field</td>
<td>WinCC Online-TrendControl</td>
<td></td>
<td>WinCC UserArchiveControl</td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>Faceplate Instance</td>
<td>WinCC OnlineTableControl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polygon</td>
<td>Graphic object</td>
<td>WinCC push button control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polyline</td>
<td>Combo box</td>
<td>WinCC SliderControl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rectangle</td>
<td>List box</td>
<td>WinCC slider control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rounded rectangle</td>
<td>Multiple row text</td>
<td>WinCC UserArchiveControl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connector</td>
<td>OLE object</td>
<td>Group display</td>
<td>Text list</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Status display</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Detailed descriptions of the individual object types is provided under “ScreenItem Object Types”. The ScreenItem object's “Type” property can be used to address the object types via the VBS Type ID.

Object properties

The "ScreenItem" object has different properties according to the features. The following section describes the properties which all ScreenItem object types have:

```vbs
Sub OnClick(ByVal Item)
Dim obj
Set obj = ScreenItems("Circle1")
End Sub
```

When a specific object type is addressed, certain further properties are added to the standard properties:

```vbs
Sub OnClick(ByVal Item)
Dim obj
Set obj = ScreenItems("Circle1")
obj.
End Sub
```

The additional properties are indicated in the descriptions of the individual object types.
Example

In the following example, the radius of a circle is set to 2 in Runtime per mouse click:

Sub OnClick(ByVal Item)
    'VBS5
    Dim objCircle
    Set objCircle = ScreenItems("Circle1")
    objCircle.Radius = 2
End Sub

See also

Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
Parent Property (Page 529)
Left Property (Page 476)
Layer Property (Page 460)
Height Property (Page 443)
Enabled Property (Page 408)
Activate Method (Page 712)
Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
Properties (Page 301)
Objects and Lists (Page 107)
Object types of the ScreenItem object (Page 143)
Object types of the ScreenItem object (Page 143)
### ScreenItems Object (List)

#### Description

The "ScreenItems" list can be used to reference an object in the picture.

#### Parent Object

Picture containing the picture element.

#### Usage

The "ScreenItems" list can be used to:

- To display or edit all objects in the list (i.e. all objects within a picture) ("_NewEnum" property).
- To count the objects in a picture ("Count" property).
- To process a specific object in the list ("Item" method).

The properties are standard properties and methods of a collection and are not described in detail in the WinCC documentation.

#### Special features of the ScreenItem object

If an external control (ActiveX control or OLE object) is embedded in WinCC, it is possible that the properties of the embedded controls have the same name with the general properties of the ScreenItem object. In such cases, the ScreenItem properties have priority.

The properties of the embedded controls can also be addressed via the "object" property:

The "object" property is only provided by ActiveX controls and OLE objects.
Example:

'Control1 is an embedded ActiveX-Control with property "type"
'VBS196
Dim Control
Set Control=ScreenItems("Control1")
Control.object.type

'Control1 is a WinCC-Control
'VBS197
Dim Control
Set Control=ScreenItems("Control1")
Control.type

Example

In the following example, the name of the objects in the current picture are displayed in a message box:

Sub OnClick(ByVal Item)
'VBS6
    Dim lngAnswer
    Dim lngIndex
    lngIndex = 1
    For lngIndex = 1 To ScreenItems.Count
        lngAnswer = MsgBox(ScreenItems(lngIndex).Objectname, vbOKCancel)
        If vbCancel = lngAnswer Then Exit For
    Next
End Sub

See also

Count Property (Page 393)
Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
ScreenItem Object (Page 125)
Parent Property (Page 529)
Item Method (Page 770)
Object types of the ScreenItem object (Page 143)
1.14.2.18 Screen Object

Description

The Screen object returns the result of access to the Screen list. All the properties and methods of this object can also be edited directly in Runtime. The "Screen" object represents a WinCC picture in Runtime and contains all the properties of the picture document and picture view.

The "Screen" object also contains the following:

- A list of all the graphic objects contained in the addressed picture which can be addressed by the "ScreenItems" object.
- A list of all the layers contained in the addressed picture which can be addressed by the "Layers" object.

Parent Object

A picture window in which the Screen object is embedded.

When the Screen object is the basic picture, the Parent object is not defined and set to zero.

Usage

The "Screen" object can be used for the following, for example:

- To release or block the operation of a screen ("Enabled" property).
- Change the width and height of a screen ("Height" and "Width" properties).
- Zoom a picture ("Zoom" property).
- Modify the fill pattern, background color and fill pattern color ("Fillstyle", "Backcolor" and "Fillcolor" properties).

**Note**

If a Change Picture is executed, all the open references are invalid for pictures no longer open. It is then no longer possible to work with these references.

**Example:**

In the following example, the width of the first picture in Runtime is increased by 20 pixels:

'VBS7
Dim objScreen
Set objScreen = HMIRuntime.Screens(1)
MsgBox "Screen width before changing: " & objScreen.Width
objScreen.Width = objScreen.Width + 20
MsgBox "Screen width after changing: " & objScreen.Width

**Notes on Cross References**

All the pictures which are addressed with the standard formulation

HMIRuntime.BaseScreenName = "Screenname"

are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.

If pictures are addressed with different formulations in the code, this can be notified by the following section of the CrossReference:

' ' WINCC:SCREENNAME_SECTION_START
Const ScreenNameInAction = "ScreenName"
' WINCC:SCREENNAME_SECTION_END
The section can be inserted in VBS actions as often as required.

**Note**

Always enter picture names without the extension "PDL" for reasons of compatibility with future versions.

**See also**

ScreenItems Property (Page 561)
Refresh Method (Page 788)
Activate Method (Page 712)
Zoom Property (Page 709)
Width Property (Page 698)
Parent Property (Page 529)
ObjectSizeDeclutteringMin Property (Page 515)
ObjectSizeDeclutteringMax Property (Page 514)
ObjectSizeDeclutteringEnable Property (Page 513)
ObjectName Property (Page 512)
Layers Property (Page 476)
DataSet Property (Page 398)
LayerDeclutteringEnable Property (Page 475)
Height Property (Page 443)
FillStyle Property (Page 419)
FillColor Property (Page 418)
ExtendedZoomingEnable Property (Page 416)
Enabled Property (Page 408)
BackColor Property (Page 322)
ActiveScreenItem Property (Page 304)
AccessPath Property (Page 302)
1.14.2.19 Screens Object (List)

Description

By using the picture window technique, several windows can be opened simultaneously in WinCC Runtime but only one basic picture exists. The "Screens" list enables access to all open pictures in Runtime using the picture names. The Screens list contains all invisible pictures.

Usage

When configuring a multi-user project, it is essential to specify the server prefix to access a picture which is not on the local computer.

The "Screens" list can be used to:

- Display or edit all the pictures within the list ("_NewEnum" property).
- To count the pictures in a project ("Count" property).
- To process a specific picture in the list ("Item" method).
- Initiate new drawing of all visible pictures ("Refresh" method).

The properties are standard properties and methods of a collection and are not described in detail in the WinCC documentation.

The access code, required in the VBS environment in the HMIRuntime.Screens(<Zugriffsschlüssel>) instruction, must fulfill the syntax requirements:

```
[<Grundbildname>.[]<Bildfenstername>[::<Bildname>]] ...
.<Bildfenstername>[::<Bildname>]
```
This means:

- The access code expresses the picture hierarchy.
- The picture names in the code can be omitted at any point.
- The “AccessPath” property of the “Screen” object corresponds to the full access code.
- Always enter picture names without the extension “PDL” for reasons of compatibility with future versions.
- The basic picture can be addressed by the access code "."

In addition, it has been defined that the basic picture can be addressed with Index 1.

Examples

The pictures are addressed by the hierarchy information in the list. There are two options here, with or without use of the picture name. In the following examples, a basic picture “BaseScreenName” is configured with a picture window “ScreenWindow”. The picture window contains the picture "ScreenName".

**Addressing with the picture name**

'VBS8
Set objScreen = HMIRuntime.Screens("BaseScreenName.ScreenWindow:ScreenName")

**Addressing without the picture name**

'VBS9
Set objScreen = HMIRuntime.Screens("ScreenWindow")

**Referencing the basic picture in various ways**

'VBS10
Set objScreen = HMIRuntime.Screens(1)

'VBS11
Set objScreen = HMIRuntime.Screens("")

'VBS12
Set objScreen = HMIRuntime.Screens("BaseScreenName")

See also

ScreenItem Object (Page 125)
Refresh Method (Page 788)
1.14.2.20 SmartTags Object

Description

The "HMIRuntime" component was deactivated in the faceplate type. The new "SmartTags" component was added for the faceplate type. With the SmartTags object you can dynamize the faceplate type. You can only access the faceplate variables and the properties of the faceplate type. You cannot access the normal WinCC tag management system. The normal WinCC tag management system is not available in the faceplate type.

Usage

Using the "SmartTags" object, you can:

- Access the faceplate tags in a faceplate type.
  Syntax: `SmartTags("<tagname>")`
- Access the properties of a faceplate type.
  Syntax: `SmartTags("Properties\<propertyname>")`

Example 1

Insert a rectangle and a button in a faceplate type. Define a faceplate variable var1. Connect the "Width" property of the rectangle to faceplate variable var1. Dynamize the "OnClick" event of the button as follows with VBS.

```vbs
'VBS306
Dim w
w = SmartTags("var1")
w = w + 10
SmartTags("var1") = w
```

When you activate Runtime, the faceplate variable is incremented by 10 every time you click the button. This increases the rectangle width by 10.

Direct reading and writing with object reference

In the following example, the SmartTags object is used to create an object reference "w" to "var1".

Referencing offers the advantage of being able to access the "var1" tag.

```vbs
'VBS307
Dim w
Set w = SmartTags("var1")
w.value = w.value + 10
```
Example 2:

Insert a rectangle and a button in a faceplate type. Define the instance-specific property "wide". Link the "Width" property of the rectangle to the instance-specific property "wide". Dynamize the "OnClick" event of the button as follows with VBS:

```vbs
'VBS308
Dim w
w = SmartTags("Properties\wide")
SmartTags("Properties\wide") = w + 50
```

When you activate Runtime, the instance-specific property "wide" is increased by 50 every time you click the button. This increases the rectangle width by 50.

See also

SmartTag property (Page 580)

1.14.2.21 Tag Object

Description

A tag object is returned via the "Tags" list. A tag object can be used to address all the properties and methods of a tag.

When creating a tag object, all the properties are installed with the following values:

- Value = VT_EMPTY
- Name = Tag name
- QualityCode = BAD NON-SPECIFIC
- TimeStamp = 0
- LastError = 0
- ErrorDescription = ""

Note

A summary of possible Quality Codes may be found in WinCC Information System under key word "Communication" > "Diagnostics" or "Communication" > "Quality Codes".
Usage

The "Tag" object can be used to:

- Read out information on the tag ("Name", "QualityCode", "TimeStamp", "LastError" and "ErrorDescription" properties)
- Set a value for a tag ("Write" method, "Value" property)
- Read a value for a tag ("Read" method, "Value" property)

Read the value of a "Tag1" tag:

'VBS13
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read()
MsgBox objTag.Value

Declaration of tags in WinCC

Always define internal tags in VB script using the "Dim" instruction in order to prevent writing tags wrongly.

When creating a new action, the "Option explicit" instruction is automatically entered in the declaration area and cannot be deleted.

Do not use the "Option explicit" instruction in the code because it may cause Runtime errors.

Example: Declaration of a VBScript "lngVar" tag:

'VBS14
Dim lngVar
lngVar = 5
MsgBox lngVar

Note

Tag names must not contain any special characters.

Please note that when creating a tag, it must not contain a value (Value = VT_EMPT0). Initialize the tags after declaration with the corresponding value.

Notes on Cross References

All the pictures which are addressed with the standard formulation

HMIRuntime.Tags("Tagname")
are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.
If tags are addressed with different formulations in the code, this can be notified by the following section of the CrossReference:

```
' ' WINCC:TAGNAME_SECTION_START
Const TagNameInAction = "TagName"
' WINCC:TAGNAME_SECTION_END
```

The section can be inserted in VBS actions as often as required.

**Note**

It is not possible to guarantee the compilation of combined tag names from the CrossReference.

**See also**

Name Property (Page 509)
Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
Write Method (Page 813)
Read Method (Page 783)
Value Property (Page 680)
TimeStamp Property (Page 622)
QualityCode Property (Page 546)
LastError Property (Page 459)
ErrorDescription Property (Page 411)
1.14.2.22 Tags Object (List)

Description

The "Tags" list enables access to tags in WinCC Runtime. The result of access to the "Tags" list is returned by an object of the type "Tag". The Tag object can be used to access all the tag properties and methods.

Note

"Tags" is a list with a restricted functional scope. The tags in the list cannot be accessed via the index but only by using the tag names. The standard methods get_Count and get_NewEnum cannot be used in the Tags list.

Usage

Tags in the list are accessed via:

HMIRuntime.Tags("Tagname")

The Tags list is used to declare tags (tag objects) for read and write access. To ensure that read and write access is carried out without errors, the corresponding tags must be available in WinCC tag management.

In VBS you can address tags directly via the name and set and read values. If you want to access additional tag properties, request the quality code, for example, you will always have to address tags via the tag listing. The tag object returned enables access to all tag properties and methods. You have to form an instance for the object, to write a binary tag with HMIRuntime.Tags("Variable").Value=TRUE, for example.

The "CreateTagSet" method can be used to generate a "TagSet" object that enables simultaneous access to several tags.

Example:

There are two options when creating tags:

- With specification of the server prefix: For tags in multi-user systems which are not stored locally.
- Direct use of the tag name: For tags stored locally on the computer.
Specification of the server prefix

'VBS15
Dim objTag
Set objTag = HMIRuntime.Tags("Serverprefix::Tagname")
If the server prefix is entered directly, the "ServerPrefix" property is assigned the corresponding value.

Specification of the tag name

'VBS16
Dim objTag
Set objTag = HMIRuntime.Tags("Tagname")
If just the tag name is used, the "ServerPrefix" and "TagPrefix" properties are assigned the values from the current context (current picture window).

See also

Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
Item Method (Page 770)
CreateTagSet Method (Page 717)
Tag Object (Page 137)

1.14.2.23 TagSet Object (List)

Description

The object "TagSet" enables simultaneous access to several tags in one call. This features better performance and lower communication load that single access to various tags.

Usage

Using the TagSet object, you may:
- Add tags to the list ("Add" method)
- Access tag objects contained in the list, and their properties ("Item" method)
- Write all tags of the list ("Write" method)
- Read all tags of the list ("Read" method)
- Remove single tags from the list ("Remove" method)
- Remove all tags from the list ("RemoveAll" method)
Tags in the list are accessed via:

'DBS169
Dim myTags
myTags = HMIRuntime.Tags.CreateTagSet
myTags("Tagname")

In order to have error-free read/write access to tags (tag objects) of the list, the respective tags must exist in WinCC tag management.

If an error occurred during read/write access, the method used will return an error message using the "LastError" and "ErrorDescription" properties.

Synchronous writing and reading of the tags is possible. The optional "Writemode" parameter can be used to write process tags directly to the AS with "1", for example, "group.Write 1". Use the optional "Readmode" parameter to read process tags with "1" directly from the AS or channel, for example, "group.Read 1".

Example:

The following example shows how to generate a TagSet object, how to add tags, and how to write values.

'VBS168
Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
'Add Tags to the Collection
group.Add "Motor1"
group.Add "Motor2"
'Add the Values of the Tags
group("Motor1").Value = 3
group("Motor2").Value = 9
'Write the Values to the DataManager
group.Write

See also

LastError Property (Page 459)
Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
Write Method (Page 813)
RemoveAll Method (Page 792)
Remove Method (Page 789)
Read Method (Page 783)
Item Method (Page 770)
1.14.3 Object types of the ScreenItem object

1.14.3.1 Object types of the ScreenItem object

Introduction

The following section lists all the available types of the "ScreenItem" object.

The features of the "ScreenItem" object represent all the graphic objects available in WinCC Graphics Designer.

The object types are divided into the following groups according to their arrangement in Graphics Designer:

- Standard objects
- Smart objects
- Windows objects
- Tube objects
- Controls

There are also the object types

- Customized Object
- Group

See also

ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Group (Page 300)
Customized Object (Page 298)
Controls (Page 218)
1.14.3.2 Standard objects

Ellipse

Description

Object Type of ScreenItem Object. Represents the graphic object "Ellipse"

Type Identifier in VBS

HMIEllipse

Usage

In the following example, the object with the name "Ellipse1" is moved 10 pixels to the right:

'VBS17
Dim objEllipse
Set objEllipse = ScreenItems("Ellipse1")
objEllipse.Left = objEllipse.Left + 10

See also

FillStyle Property (Page 419)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
RadiusWidth Property (Page 548)
RadiusHeight Property (Page 547)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BackFlashColorOn Property (Page 324)
BackFlashColorOff Property (Page 324)
BackColor Property (Page 322)
Layer Property (Page 460)
Ellipse arc

Description

Object Type of ScreenItem Object. Represents the graphic object "Ellipse Arc"

Type Identifier in VBS

HMIEllipticalArc

Usage

In the following example, the object with the name "EllipseArc1" is moved 10 pixels to the right:

'VBS18
Dim objEllipseArc
Set objEllipseArc = ScreenItems("EllipseArc1")
objEllipseArc.Left = objEllipseArc.Left + 10

See also

RadiusHeight Property (Page 547)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
StartAngle Property (Page 587)
Ellipse segment

Description

Object Type of ScreenItem Object. Represents the graphic object "Ellipse Segment"

Type Identifier in VBS

HMIEllipseSegment
Usage

In the following example, the object with the name "EllipseSegment1" is moved 10 pixels to the right:

'VBS19
Dim objEllipseSeg
Set objEllipseSeg = ScreenItems("EllipseSegment1")
objEllipseSeg.Left = objEllipseSeg.Left + 10

See also

Layer Object (Page 120)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
StartAngle Property (Page 587)
RadiusWidth Property (Page 548)
RadiusHeight Property (Page 547)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
Circle

Description

Object Type of ScreenItem Object. Represents the graphic object "Circle".

Type Identifier in VBS

HMICircle

Usage

In the following example, the object with the name "Circle1" is moved 10 pixels to the right:

```vbs
' VBS20
Dim objCircle
Set objCircle = ScreenItems("Circle1")
objCircle.Left = objCircle.Left + 10
```
See also

Properties (Page 301)
BorderStyle Property (Page 355)
Activate Method (Page 712)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Radius Property (Page 547)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BackFlashColorOn Property (Page 324)
Circular arc

Description

Object Type of ScreenItem Object. Represents the graphic object "Circular Arc"

Type Identifier in VBS

HMI_CircularArc

Usage

In the following example, the object with the name "CircularArc1" is moved 10 pixels to the right:

'VBS21
Dim objCircularArc
Set objCircularArc = ScreenItems("CircularArc1")
objCircularArc.Left = objCircularArc.Left + 10

See also

StartAngle Property (Page 587)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Pie segment

Description

Object Type of ScreenItem Object. Represents the graphic object "Pie Segment"
Type Identifier in VBS

HMICircleSegment

Usage

In the following example, the object with the name "PieSegment1" is moved 10 pixels to the right:

```vbs
'VBS2
Dim objCircleSeg
Set objCircleSeg = ScreenItems("PieSegment1")
objCircleSeg.Left = objCircleSeg.Left + 10
```

See also

Type Property (Page 666)
BorderColor Property (Page 353)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Top Property (Page 643)
ToolTipText Property (Page 641)
StartAngle Property (Page 587)
Radius Property (Page 547)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
Line

Description

Object Type of ScreenItem Object. Represents the graphic object "Line"

Type Identifier in VBS

HMILine
Usage

In the following example, the object with the name "Line1" is moved 10 pixels to the right:

```vbs
'VBS23
Dim objLine
Set objLine = ScreenItems("Line1")
objLine.Left = objLine.Left + 10
```

See also

PasswordLevel Property (Page 530)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
RotationAngle Property (Page 552)
ReferenceRotationTop Property (Page 549)
ReferenceRotationLeft Property (Page 549)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashBorderColor Property (Page 424)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderEndStyle Property (Page 354)
Polygon

Description

Object Type of ScreenItem Object. Represents the graphic object "Polygon"

Type Identifier in VBS

HMIPolygon

Usage

In the following example, the object with the name "Polygon1" is moved 10 pixels to the right:

```
'VBS24
Dim objPolygon
Set objPolygon = ScreenItems("Polygon1")
objPolygon.Left = objPolygon.Left + 10
```

See also

ReferenceRotationTop Property (Page 549)
BackFlashColorOn Property (Page 324)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
RotationAngle Property (Page 552)
ReferenceRotationLeft Property (Page 549)
PointCount Property (Page 541)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Index Property (Page 452)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BackFlashColorOff Property (Page 324)
BackColor Property (Page 322)
ActualPointTop Property (Page 306)
ActualPointLeft Property (Page 305)
Layer Property (Page 460)
Polyline

Description

Object Type of ScreenItem Object. Represents the graphic object "Polyline"

Type Identifier in VBS

HMIPolyLine

Usage

In the following example, the object with the name "Polyline1" is moved 10 pixels to the right:

'VBS25
Dim objPolyline
Set objPolyline = ScreenItems("Polyline1")
objPolyline.Left = objPolyline.Left + 10

See also

Layer Object (Page 120)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
RotationAngle Property (Page 552)
ReferenceRotationTop Property (Page 549)  
ReferenceRotationLeft Property (Page 549)  
PointCount Property (Page 541)  
PasswordLevel Property (Page 530)  
Parent Property (Page 529)  
ObjectName Property (Page 512)  
Left Property (Page 476)  
Index Property (Page 452)  
Height Property (Page 443)  
FlashRateBorderColor Property (Page 427)  
FlashBorderColor Property (Page 424)  
Enabled Property (Page 408)  
BorderWidth Property (Page 355)  
BorderStyle Property (Page 355)  
BorderFlashColorOn Property (Page 355)  
BorderFlashColorOff Property (Page 354)  
BorderEndStyle Property (Page 354)  
BorderColor Property (Page 353)  
BorderBackColor Property (Page 353)  
ActualPointTop Property (Page 306)  
ActualPointLeft Property (Page 305)  
Layer Property (Page 460)

**Rectangle**

**Description**

Object Type of ScreenItem Object. Represents the graphic object "Rectangle"
Type Identifier in VBS

HMIRectangle

Usage

In the following example, the object with the name "Rectangle1" is moved 10 pixels to the right:

'VBS26
Dim objRectangle
Set objRectangle = ScreenItems("Rectangle1")
objRectangle.Left = objRectangle.Left + 10

Notes on Error Handling

The rectangle and rounded rectangle are mapped to an "HMIRectangle" type in the object model. Since the two objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handing. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.
Examples of error handling

Sub OnClick(ByVal Item)
  'VBS27
  Dim objScreenItem
  'Activation of error handling:
  On Error Resume Next
  For Each objScreenItem In ScreenItems
    If "HMIRectangle" = objScreenItem.Type Then
      '=== Property "RoundCornerHeight" only available for RoundedRectangle
      objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 2
      If 0 <> Err.Number Then
        HMIRuntime.Trace objScreenItem.Name & ": no RoundedRectangle" & vbCrLf
      End If
  End If
  'Delete error message
  Err.Clear
  End If
  Next
  On Error Goto 0  'Deactivation of error handling
End Sub

See also

Properties (Page 301)
BorderFlashColorOn Property (Page 355)
Activate Method (Page 712)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
Rounded rectangle

Description

Object Type of ScreenItem Object. Represents the graphic object "Rounded Rectangle".

Type Identifier in VBS

HMIRoundRectangle
Usage

In the following example, the object with the name "RoundedRectangle1" is moved 10 pixels to the right:

'VBScript
Dim objRoundedRectangle
Set objRoundedRectangle = ScreenItems("RoundedRectangle1")
objRoundedRectangle.Left = objRoundedRectangle.Left + 10

Notes on Error Handling

The rectangle and rounded rectangle are mapped to an "HMIRectangle" type in the object model. Since the two objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handling. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.
Examples of error handling

Sub OnClick(ByVal Item)
    'VBS29
    Dim objScreenItem
    On Error Resume Next  'Activation of errorhandling
    For Each objScreenItem In ScreenItems
        If "HMIRectangle" = objScreenItem.Type Then
            '=== Property "RoundCornerHeight" available only for RoundRectangle
            objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 2
            If 0 <> Err.Number Then
                HMIRuntime.Trace objScreenItem.ObjectName & ": no RoundedRectangle" & vbCrLf
                Err.Clear    'Delete errormessage
            End If
        End If
    Next
    On Error Goto 0    'Deactivation of errorhandling
End Sub

See also

- FlashBackColor Property (Page 423)
- Activate Method (Page 712)
- Properties (Page 301)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Width Property (Page 698)
- Visible Property (Page 696)
- Type Property (Page 666)
- Top Property (Page 643)
- ToolTipText Property (Page 641)
- RoundCornerWidth Property (Page 553)
- RoundCornerHeight Property (Page 553)
- PasswordLevel Property (Page 530)
- Parent Property (Page 529)
- ObjectName Property (Page 512)
- Left Property (Page 476)
- Layer Object (Page 120)
- Height Property (Page 443)
- FlashRateBorderColor Property (Page 427)
Object Type of ScreenItem Object. Represents the graphic object "Static Text"

Type Identifier in VBS

HMITextField
Usage

In the following example, the object with the name "StaticText1" is moved 10 pixels to the right:

```vbs
'VBS30
Dim objStaticText
Set objStaticText = ScreenItems("StaticText1")
objStaticText.Left = objStaticText.Left + 10
```

See also

ObjectName Property (Page 512)
BorderFlashColorOn Property (Page 355)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Text list (Page 196)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
Orientation Property (Page 526)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
ForeFlashColorOn Property (Page 436)
ForeFlashColorOff Property (Page 435)
ForeColor Property (Page 435)
FontUnderline Property (Page 434)
FontSize Property (Page 433)
FontName Property (Page 433)
FontItalic Property (Page 432)
FontBold Property (Page 432)
Object Type of ScreenItem Object. Represents the graphic object "Connector"
Type Identifier in VBS

HMIConnector

Usage

In the following example, the object with the name "Connector1" is moved 10 pixels to the right:

```
'VBS31
Dim objConnector
Set objConnector = ScreenItems("Connector1")
objConnector.Left = objConnector.Left + 10
```

See also

ScreenItems Object (List) (Page 129)
Activate Method (Page 712)
Properties (Page 301)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
TopConnectedObjectName Property (Page 644)
TopConnectedConnectionPointIndex Property (Page 644)
Top Property (Page 643)
ToolTipText Property (Page 641)
Parent Property (Page 529)
Orientation Property (Page 526)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Property (Page 460)
Height Property (Page 443)
Enabled Property (Page 408)
BottomConnectedObjectName Property (Page 356)
BottomConnectedConnectionPointIndex Property (Page 356)
1.14.3.3 Smart objects

3D Bar

Description

Object Type of ScreenItem Object. Represents the graphic object “3D Bar”

Type Identifier in VBS
HMIBar

Usage

In the following example, the object with the name “3DBar1” is moved 10 pixels to the right:

'VBS32
Dim objBar
Set objBar = ScreenItems("3DBar1")
objBar.Left = objBar.Left + 10

Notes on Error Handling

Bars and 3D bars are imaged in the object model on a "HMIBar" type. Since the two objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.
The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

```
On Error Goto 0
```

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handing. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.

### Examples of error handling

```
'VBS148
Sub OnClick(ByVal Item)
Dim objScreenItem
'
'Activation of errorhandling:
On Error Resume Next
For Each objScreenItem In ScreenItems
If "HMIBar" = objScreenItem.Type Then
'=== Property "Layer00Value" only available for 3D bar
objScreenItem.Layer00Value = objScreenItem.Layer00Value * 2
If 0 <> Err.Number Then
HMIRuntime.Trace objScreenItem.Name & ": no 3D bar" & vbCrLf
' 'Delete error message
Err.Clear
End If
End If
Next
On Error Goto 0 'Deactivation of errorhandling
End Sub
```

### See also

- Type Property (Page 666)
- Layer08Color Property (Page 467)
BorderStyle Property (Page 355)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
ZeroPointValue Property (Page 709)
Width Property (Page 698)
Visible Property (Page 696)
Top Property (Page 643)
ToolTipText Property (Page 641)
Process Property (Page 544)
PredefinedAngles Property (Page 543)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Min Property (Page 506)
Max Property (Page 488)
LightEffect Property (Page 477)
Left Property (Page 476)
Layer10Value Property (Page 475)
Layer09Value Property (Page 475)
Layer08Value Property (Page 475)
Layer07Value Property (Page 474)
Layer06Value Property (Page 474)
Layer05Value Property (Page 474)
Layer04Value Property (Page 473)
Layer03Value Property (Page 473)
Layer02Value Property (Page 473)
Layer01Value Property (Page 473)
Layer00Value Property (Page 472)
Layer10Color Property (Page 468)
Layer09Color Property (Page 468)
Layer07Color Property (Page 467)
Layer06Color Property (Page 467)
Layer05Color Property (Page 466)
Layer04Color Property (Page 466)
Layer03Color Property (Page 466)
Layer02Color Property (Page 465)
Layer01Color Property (Page 465)
Layer00Color Property (Page 465)
Layer10Checked Property (Page 464)
Layer09Checked Property (Page 464)
Layer08Checked Property (Page 464)
Layer07Checked Property (Page 463)
Layer06Checked Property (Page 463)
Layer05Checked Property (Page 463)
Layer04Checked Property (Page 462)
Layer03Checked Property (Page 462)
Layer02Checked Property (Page 462)
Layer01Checked Property (Page 461)
Layer00Checked Property (Page 461)
Layer Object (Page 120)
Height Property (Page 443)
Enabled Property (Page 408)
Direction Property (Page 405)
BorderWidth Property (Page 355)
BorderColor Property (Page 353)
BaseY Property (Page 342)
BaseX Property (Page 343)
BarWidth Property (Page 340)
BarHeight Property (Page 329)
BarDepth Property (Page 328)
Background Property (Page 324)
BackColor Property (Page 322)
Axe Property (Page 320)
AngleBeta Property (Page 312)
AngleAlpha Property (Page 312)
Application Window

Description

Object Type of ScreenItem Object. Represents the graphic object "Application Window"

Type Identifier in VBS

HMIApplicationWindow

Usage

In the following example, the object with the name "ApplicationWindow1" is moved 10 pixels to the right:

'VBS33
Dim objAppWindow
Set objAppWindow = ScreenItems("ApplicationWindow1")
objAppWindow.Left = objAppWindow.Left + 10

See also

Properties (Page 301)
Activate Method (Page 712)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
WindowBorder Property (Page 699)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
Template Property (Page 598)
Parent Property (Page 529)
OnTop Property (Page 517)
ObjectName Property (Page 512)
Moveable Property (Page 507)
MaximizeButton Property (Page 488)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
Enabled Property (Page 408)
CloseButton Property (Page 371)
Caption Property (Page 362)
Application Property (Page 313)

Bar

Description

Object Type of ScreenItem Object. Represents the graphic object "Bar"

Type Identifier in VBS

HMIBar

Usage

In the following example, the object with the name "Bar1" is moved 10 pixels to the right:

'VBS34
Dim objBar
Set objBar = ScreenItems("Bar1")
objBar.Left = objBar.Left + 10
Notes on Error Handling

Bars and 3D bars are imaged in the object model on a "HMIBar" type. Since the two objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handling. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.
Examples of error handling

'VBS147
Sub OnClick(ByVal Item)
Dim objScreenItem

'Activation of error handling:
On Error Resume Next
For Each objScreenItem In ScreenItems
If "HMIBar" = objScreenItem.Type Then
  '=== Property "LimitHigh4" only available for bar
  objScreenItem.LimitHigh4 = objScreenItem.LimitHigh4 * 2
  If 0 <> Err.Number Then
    HMIRuntime.Trace objScreenItem.Name & " no bar" & vbCrLf
  End If
  'Delete error message
  Err.Clear
End If
Next
On Error Goto 0  'Deactivation of error handling
End Sub

See also

TooltipText Property (Page 641)
Layer Object (Page 120)
ColorChangeType Property (Page 375)
Average Property (Page 320)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
ZeroPointValue Property (Page 709)
ZeroPoint Property (Page 709)
Width Property (Page 698)
WarningLow Property (Page 698)
WarningHigh Property (Page 697)
Visible Property (Page 696)
TypeWarningLow Property (Page 671)
TypeWarningHigh Property (Page 670)
TypeToleranceLow Property (Page 670)
TypeToleranceHigh Property (Page 670)
TypeLimitLow5 Property (Page 670)
TypeLimitLow4 Property (Page 669)
TypeLimitHigh5 Property (Page 669)
TypeLimitHigh4 Property (Page 669)
TypeAlarmLow Property (Page 668)
TypeAlarmHigh Property (Page 668)
Type Property (Page 666)
LTrendColor property (before WinCC V7) (Page 647)
Trend Property (Page 645)
Top Property (Page 643)
ToleranceLow Property (Page 628)
ToleranceHigh Property (Page 628)
ScalingType Property (Page 558)
Scaling Property (Page 558)
ScaleTicks Property (Page 558)
ScaleColor Property (Page 558)
RightComma Property (Page 551)
Process Property (Page 544)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Min Property (Page 506)
Max Property (Page 488)
Marker Property (Page 488)
LongStrokesTextEach Property (Page 485)
LongStrokesSize Property (Page 485)
LongStrokesOnly Property (Page 485)
LongStrokesBold Property (Page 484)
LimitLow5 Property (Page 479)
LimitLow4 Property (Page 478)
LimitHigh5 Property (Page 478)
LimitHigh4 Property (Page 478)
LeftComma Property (Page 477)
Left Property (Page 476)
HysteresisRange Property (Page 451)
Hysteresis Property (Page 451)
Height Property (Page 443)
FontSize Property (Page 433)
FontName Property (Page 433)
FontBold Property (Page 432)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle2 Property (Page 421)
FillStyle Property (Page 419)
FillColor Property (Page 418)
Exponent Property (Page 412)
Enabled Property (Page 408)
Direction Property (Page 405)
ColorWarningLow Property (Page 378)
ColorWarningHigh Property (Page 378)
ColorToleranceLow Property (Page 377)
ColorToleranceHigh Property (Page 377)
ColorLimitLow5 Property (Page 377)
ColorLimitLow4 Property (Page 376)
ColorLimitHigh5 Property (Page 376)
ColorLimitHigh4 Property (Page 376)
ColorAlarmLow Property (Page 375)
ColorAlarmHigh Property (Page 374)
CheckWarningLow Property (Page 370)
CheckWarningHigh Property (Page 369)
CheckToleranceLow Property (Page 369)
CheckToleranceHigh Property (Page 369)
CheckLimitLow5 Property (Page 368)
CheckLimitLow4 Property (Page 368)
CheckLimitHigh5 Property (Page 368)
CheckLimitHigh4 Property (Page 367)
CheckAlarmLow Property (Page 367)
Picture Window

Description

Object Type of ScreenItem Object. Represents the graphic object “Picture Window”

Type Identifier in VBS

HMIScreenWindow
Usage

In the following example, the object with the name "ScreenWindow1" is moved 10 pixels to the right:

'VBS35
Dim objScrWindow
Set objScrWindow = ScreenItems("ScreenWindow1")
objScrWindow.Left = objScrWindow.Left + 10

See also

ServerPrefix Property (Page 571)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Zoom Property (Page 709)
WindowBorder Property (Page 699)
Width Property (Page 698)
Visible Property (Page 696)
UpdateCycle Property (Page 673)
Type Property (Page 666)
Top Property (Page 643)
TagPrefix Property (Page 596)
ScrollPositionY Property (Page 562)
ScrollPositionX Property (Page 562)
ScrollBars Property (Page 561)
ScreenName Property (Page 560)
Screens Property (Page 560)
Parent Property (Page 529)
OnTop Property (Page 517)
OffsetTop Property (Page 516)
OffsetLeft Property (Page 515)
ObjectName Property (Page 512)
Moveable Property (Page 507)
MaximizeButton Property (Page 488)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
Enabled Property (Page 408)
CloseButton Property (Page 371)
CaptionText Property (Page 364)
Caption Property (Page 362)
AdaptSize Property (Page 307)
AdaptPicture Property (Page 307)
MenuToolBarConfig Property (Page 495)

Control

Description

Object Type of ScreenItem Object. Represents the graphic object "Control"

The Control object type always assumes the properties of the Control type selected. In the case of controls provided by WinCC, the properties are indicated under the description of the corresponding Control.

In the case of controls from external suppliers, the control properties are supplied and thus not a part of this description. However, the control properties can be queried using the "Item" property.

Type Identifier in VBS

Special WinCC type descriptions or version-independent ProgID
Usage

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

'VBS36
Dim objControl
Set objControl = ScreenItems("Controll")
objControl.Left = objControl.Left + 10

Special feature

The controls provided by WinCC return a special ID as the type. It can be found under the topic "Type Identification in VBS" in the individual descriptions of the WinCC Controls.

Use of Controls from External Suppliers

In the case of non-WinCC controls, the version-independent ProgID is returned as the type.

It is possible to determine the version-dependent ProgID or "User friendly Name" from the ProgID: In the following example, "Control1" is a control embedded in the picture which already returns the version-independent ProgID as a result of the Type property.

Note

Since not every Control has a version-dependent ProgID, an error handling measure should be integrated to query the version-dependent ProgID or UserFriendlyName. If no error handling is used, the code is terminated immediately without any result when no ProgID is found.

Determine the version-dependent ProgID as follows:

'VBS37
Dim objControl
Dim strCurrentVersion
Set objControl = ScreenItems("Controll")
strCurrentVersion = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & "\CurVer\")
MsgBox strCurrentVersion

Note

In order that example above works, a multimedia control should be inserted in the picture.
Determine the UserFriendlyName as follows:

```vbs
'VBS38
Dim objControl
Dim strFriendlyName
Set objControl = ScreenItems("Control1")
strFriendlyName = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & "\")
MsgBox strFriendlyName
```

**Note**

In order that example above works, a multimedia control should be inserted in the picture.

If a non-WinCC control is used, it is possible that the properties provided by the control have the same names as the general ScreenItem properties. In such cases, the ScreenItem properties have priority. The "hidden" properties of an external control supplier can be accessed using the additional "object" property. Address the properties of an external control supplier as follows:

```vbs
Control.object.type
```

The properties of the ScreenItem object are used in the case of identical names, if you use the following form:

```vbs
Control.type
```

**WinCC controls available**

- WinCC Alarm Control
- WinCC Digital/Analog Clock
- WinCC FunctionTrendControl
- WinCC Gauge Control
- WinCC Media Control
- WinCC OnlineTableControl
- WinCC OnlineTrendControl
- WinCC Push Button Control
- WinCC Slider Control
- WinCC UserArchiveControl
- HMI Symbol Library
Object Type of ScreenItem Object. Represents the graphic object "I/O Field"

Type Identifier in VBS

HMIIOField
Usage

In the following example, the object with the name "IOField1" is moved 10 pixels to the right:

```vbs
'VBS39
Dim objIOField
Set objIOField = ScreenItems("IOField1")
objIOField.Left = objIOField.Left + 10
```

See also

- OperationMessage Property (Page 517)
- EditAtOnce Property (Page 407)
- Activate Method (Page 712)
- Properties (Page 301)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Width Property (Page 698)
- Visible Property (Page 696)
- Type Property (Page 666)
- Top Property (Page 643)
- ToolTipText Property (Page 641)
- PasswordLevel Property (Page 530)
- Parent Property (Page 529)
- OutputValue Property (Page 528)
- OutputFormat Property (Page 527)
- Orientation Property (Page 526)
- OperationReport Property (Page 525)
- ObjectName Property (Page 512)
- LimitMin Property (Page 479)
- LimitMax Property (Page 479)
- Left Property (Page 476)
- Layer Object (Page 120)
- HiddenInput Property (Page 444)
- Height Property (Page 443)
- ForeFlashColorOn Property (Page 436)
- ForeFlashColorOff Property (Page 435)
Faceplate Instance

Description

Object Type of ScreenItem Object. Represents the "faceplate instance" graphic object.

Type identifier in VBS

HMIFaceplateObject

Usage

In the following example, the object with the name "FaceplateInstance1" is moved 10 pixels to the right:

```vbs
'VBS309
Dim objFaceplateObject
Set objFaceplateObject = ScreenItems("FaceplateInstance1")
objFaceplateObject.Left = objFaceplateObject.Left + 10
```

Graphic Object

Description

Object Type of ScreenItem Object. Represents the graphic object "Graphic Object"
Type Identifier in VBS

HMIGraphicView

Usage

In the following example, the object with the name "GraphicObject1" is moved 10 pixels to the right:

'VBS40
Dim objGraphicView
Set objGraphicView = ScreenItems("GraphicObject1")
objGraphicView.Left = objGraphicView.Left + 10

See also

Parent Property (Page 529)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
PicUseTransColor Property (Page 540)
PictureName Property (Page 538)
PicTransColor Property (Page 536)
PicReferenced Property (Page 535)
PasswordLevel Property (Page 530)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
Object Type of ScreenItem Object. Represents the "Combobox" graphic object.

**Type Identifier in VBS**

HMIComboBox
Usage

In the following example, the object with the name "ComboBox1" is moved 10 pixels to the right:

'VBS21
Dim objComboBox
Set objComboBox = ScreenItems("ComboBox1")
objComboBox.Left = objComboBox.Left + 10

List Box

Description

Object Type of ScreenItem Object. Represents the "List Box" graphic object.

Type Identifier in VBS

HMIListBox

Usage

In the following example, the object with the name "ListBox1" is moved 10 pixels to the right:

'VBS21
Dim objListBox
Set objListBox = ScreenItems("ListBox1")
objListBox.Left = objListBox.Left + 10
Multiple row text

Description

Object Type of ScreenItem Object. Represents the "Multiline Text" graphic object.

Type Identifier in VBS

HMIMultiLineEdit

Usage

In the following example, the object with the name "MultiLineEdit1" is moved 10 pixels to the right:

```
'VBS21
Dim objMultiLineEdit
Set objMultiLineEdit = ScreenItems("MultiLineEdit1")
objMultiLineEdit.Left = objMultiLineEdit.Left + 10
```

OLE object

Description

Object Type of ScreenItem Object. Represents the graphic object "OLE Element". The return value is a STRING type.
Type Identifier in VBS

Version-independent ProgID

Usage

In the following example, the object with the name "OLEElement1" is moved 10 pixels to the right:

'VBS41
Dim objOLEElement
Set objOLEElement = ScreenItems("OLEElement1")
objOLEElement.Left = objOLEElement.Left + 10

Special feature

In the case of OLE Elements, the version-independent ProgID is returned as the type. It is possible to determine the version-dependent ProgID or "User friendly Name" from the ProgID: In the following example, "OLEObject1" is a control embedded in the picture which already returns the version-independent ProgID as a result of the Type property.

Note

Since not every Control has a version-dependent ProgID, an error handling measure should be integrated to query the version-dependent ProgID or UserFriendlyName. If no error handling is used, the code is terminated immediately without any result when no ProgID is found.

Determine the version-dependent ProgID as follows:

'VBS42
Dim objControl
Dim strCurrentVersion
Set objControl = ScreenItems("OLEElement1")
strCurrentVersion = CreateObject("WScript.Shell").RegRead("HKCR \" & objControl.Type & \"CurVer\")
MsgBox strCurrentVersion

Note

In order that the example above works, a Word document should be embedded in the picture as an OLE Element.
Determine the User Friendly Name as follows:

'VBS43
Dim objControl
Dim strFriendlyName
Set objControl = ScreenItems("OLEElement1")
strFriendlyName = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & ")
MsgBox strFriendlyName

Note
In order that the example above works, a Word document should be embedded in the picture as an OLE Element.

Using OLE Elements

If an OLE Element is used, it is possible that the properties provided by the OLE Element have the same names as the general ScreenItem properties. In such cases, the ScreenItem properties have priority. The "hidden" properties of an OLE Element can be accessed using the additional "Object" property. Address the properties of an OLE Element as follows:

OLEObjekt.object.type

Only use the form

OLEObjekt.type

In the case of identical names, the properties of the ScreenItem object are used.

See also

Height Property (Page 443)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Group Display

Description

![Diagram of Screen hierarchy]

Object Type of ScreenItem Object. Represents the graphic object "Group Display"

Type Identifier in VBS

HMIGroupDisplay

Usage

In the following example, the object with the name "GroupDisplay1" is moved 10 pixels to the right:

```vbs
'VBS44
Dim objGroupDisplay
Set objGroupDisplay = ScreenItems("GroupDisplay1")
objGroupDisplay.Left = objGroupDisplay.Left + 10
```

See also

Activate Method (Page 712)
MCKQBackColorOn Property (Page 492)
FontBold Property (Page 432)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
UserValue4 Property (Page 678)
UserValue3 Property (Page 677)
UserValue2-Eigenschaft (Page 677)
UserValue1 Property (Page 677)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
SignificantMask Property (Page 579)
SameSize Property (Page 557)
Relevant Property (Page 550)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
MessageClass Property (Page 503)
MCText Property (Page 494)
MCKQTextFlash Property (Page 494)
MCKQTextColorOn Property (Page 493)
MCKQTextColorOff Property (Page 493)
MCKQBackFlash Property (Page 493)
MCKQBackColorOff Property (Page 492)
MCKOTextFlash Property (Page 492)
MCKOTextColorOn Property (Page 492)
MCKOTextColorOff Property (Page 491)
MCKOBackFlash Property (Page 491)
MCKOBackColorOn Property (Page 491)
MCKOBackColorOff Property (Page 490)
MCGUTextFlash Property (Page 490)
MCGUTextColorOn Property (Page 490)
MCGUTextColorOff Property (Page 490)
### MCGUBackFlash Property (Page 489)

### MCGUBackColorOn Property (Page 489)

### MCGUBackColorOff-Eigenschaft (Page 489)

### LockTextColor Property (Page 484)

### LockText Property (Page 484)

### LockStatus Property (Page 483)

### LockBackColor Property (Page 483)

### Left Property (Page 476)

### Layer Object (Page 120)

### Height Property (Page 443)

### FontUnderline Property (Page 434)

### FontSize Property (Page 433)

### FontName Property (Page 433)

### FontStyle Property (Page 432)

### FlashRate Property (Page 426)

### Enabled Property (Page 408)

### CollectValue Property (Page 373)

### Button4Width Property (Page 362)

### Button3Width Property (Page 362)

### Button2Width Property (Page 361)

### Button1Width Property (Page 361)

### BackColor Property (Page 322)

### AlignmentTop Property (Page 310)

### AlignmentLeft Property (Page 309)

---

#### Text list

<table>
<thead>
<tr>
<th>Screens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
</tr>
<tr>
<td>ScreenItems</td>
</tr>
<tr>
<td>ScreenItem</td>
</tr>
</tbody>
</table>
Object Type of ScreenItem Object. Represents the graphic object "Text List"

**Type Identifier in VBS**

HMISymbolicIOField

**Usage**

In the following example, the object with the name "TextList1" is moved 10 pixels to the right:

```vbs
'VBS45
Dim objSymIO
Set objSymIO = ScreenItems("TextList1")
objSymIO.Left = objSymIO.Left + 10
```

**See also**

- Type Property (Page 666)
- FontUnderline Property (Page 434)
- BackFlashColorOff Property (Page 324)
- Activate Method (Page 712)
- Properties (Page 301)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Width Property (Page 698)
- Visible Property (Page 696)
- UnselTextColor Property (Page 673)
- UnselBGColor Property (Page 672)
- Top Property (Page 643)
- ToolTipText Property (Page 641)
- SelTextColor Property (Page 569)
- SelBGColor Property (Page 563)
- PasswordLevel Property (Page 530)
- Parent Property (Page 529)
- OutputValue Property (Page 528)
- Orientation Property (Page 526)
- OperationReport Property (Page 525)
- OperationMessage Property (Page 517)
ObjectName Property (Page 512)
NumberLines Property (Page 511)
ListType Property (Page 481)
Left Property (Page 476)
Layer Object (Page 120)
LanguageSwitch Property (Page 458)
ItemBorderWidth Property (Page 456)
ItemBorderStyle Property (Page 455)
ItemBorderColor Property (Page 455)
ItemBorderBackColor Property (Page 455)
Height Property (Page 443)
ForeFlashColorOn Property (Page 436)
ForeFlashColorOff Property (Page 435)
ForeColor Property (Page 435)
FontSize Property (Page 433)
FontName Property (Page 433)
FontItalic Property (Page 432)
FontBold Property (Page 432)
FlashRateForeColor Property (Page 428)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashForeColor Property (Page 424)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
EditAtOnce Property (Page 407)
CursorControl Property (Page 395)
BoxType Property (Page 357)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BitNumber Property (Page 346)
BackFlashColorOn Property (Page 324)
BackColor Property (Page 322)
AssumeOnExit Property (Page 315)
Assignments Property (Page 315)
AlignmentTop Property (Page 310)
AlignmentLeft Property (Page 309)
AdaptBorder Property (Page 306)

Status display

Description

Object Type of ScreenItem Object. Represents the graphic object "Status Display"

Type Identifier in VBS

HMIGraphicIOField

Usage

In the following example, the object with the name "StatusDisplay1" is moved 10 pixels to the right:

'VBS46
Dim objGraphicIO
Set objGraphicIO= ScreenItems("StatusDisplay1")
objGraphicIO.Left = objGraphicIO.Left + 10
See also

Layer Object (Page 120)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Index Property (Page 452)
Height Property (Page 443)
FlashRateFlashPic Property (Page 428)
FlashRateBorderColor Property (Page 427)
FlashPicUseTransColor Property (Page 426)
FlashPicture Property (Page 425)
FlashPicTransColor Property (Page 425)
FlashPicReferenced Property (Page 425)
FlashFlashPicture Property (Page 424)
FlashBorderColor Property (Page 424)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BasePicUseTransColor Property (Page 341)
BasePicture Property (Page 341)
1.14.3.4 Windows objects

Button

Description

In the following example, the object with the name "Button1" is moved 10 pixels to the right:

```vbs
'VBS47
Dim cmdButton
Set cmdButton = ScreenItems("Button1")
cmdButton.Left = cmdButton.Left + 10
```

Notes on Error Handling

Buttons and pushbuttons are mapped in the object model to an "HMIButton" type. Since the objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

```vbs
On Error Resume Next
```
The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

```vbnet
On Error Goto 0
```

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handling. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.

### Examples of error handling

```vbnet
Sub OnClick(ByVal Item) 'VBS48
  Dim objScreenItem
  On Error Resume Next    'Activation of error handling
  For Each objScreenItem In ScreenItems
    If objScreenItem.Type = "HMIButton" Then
      '=== Property "Text" available only for Standard-Button
      objScreenItem.Text = "Windows"
      If 0 <> Err.Number Then
        HMIRuntime.Trace objScreenItem.ObjectName & ": no Windows-Button" & vbCrLf
        Err.Clear    'Delete error message
      End If
    End If
  Next
  On Error Goto 0    'Deactivation of error handling
End Sub
```

### See also
- Top Property (Page 643)
- FlashBorderColor Property (Page 424)
- Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
WindowsStyle Property (Page 700)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
ToolTipText Property (Page 641)
Text list (Page 196)
PictureUp Property (Page 539)
PictureDown Property (Page 537)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
Orientation Property (Page 526)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Hotkey Property (Page 450)
Height Property (Page 443)
ForeFlashColorOn Property (Page 436)
ForeFlashColorOff Property (Page 435)
ForeColor Property (Page 435)
FontUnderline Property (Page 434)
FontSize Property (Page 433)
FontName Property (Page 433)
FontItalic Property (Page 432)
FontBold Property (Page 432)
FlashRateForeColor Property (Page 428)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashForeColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColorTop Property (Page 354)
BorderColor Property (Page 353)
BorderColorBottom Property (Page 354)
BorderBackColor Property (Page 353)
BackFlashColorOn Property (Page 324)
BackFlashColorOff Property (Page 324)
BackColor Property (Page 322)
AlignmentTop Property (Page 310)
AlignmentLeft Property (Page 309)
AdaptBorder Property (Page 306)

Check box

Description

Object Type of ScreenItem Object. Represents the graphic object “Check Box”

Type Identifier in VBS

HMICheckBox
Usage

In the following example, the object with the name "CheckBox1" is moved 10 pixels to the right:

```vbs
'VBS49
Dim chkCheckBox
Set chkCheckBox = ScreenItems("CheckBox1")
chkCheckBox.Left = chkCheckBox.Left + 10
```

See also

FontSize Property (Page 433)
BackColor Property (Page 322)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Text list (Page 196)
Process Property (Page 544)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
Orientation Property (Page 526)
OperationMessage Property (Page 517)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Index Property (Page 452)
Height Property (Page 443)
ForeFlashColorOn Property (Page 436)
ForeFlashColorOff Property (Page 435)
ForeColor Property (Page 435)
FontUnderline Property (Page 434)
fontName Property (Page 433)
FontItalic Property (Page 432)
FontBold Property (Page 432)
FlashRateForeColor Property (Page 428)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashForeColor Property (Page 424)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BoxCount Property (Page 357)
BoxAlignment Property (Page 356)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BackFlashColorOn Property (Page 324)
BackFlashColorOff Property (Page 324)
AlignmentTop Property (Page 310)
AlignmentLeft Property (Page 309)
AdaptBorder Property (Page 306)
Radio box

Description

Object Type of ScreenItem Object. Represents the graphic object "Radio Box"

Type Identifier in VBS

HMIOptionGroup

Usage

In the following example, the object with the name "RadioBox1" is moved 10 pixels to the right:

```
'VBS50
Dim objOptionGroup
Set objOptionGroup = ScreenItems("RadioBox1")
objOptionGroup.Left = objOptionGroup.Left + 10
```

See also

ForeColor Property (Page 435)
BackFlashColorOn Property (Page 324)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Text list (Page 196)
Process Property (Page 544)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
Orientation Property (Page 526)
OperationMessage Property (Page 517)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Index Property (Page 452)
Height Property (Page 443)
ForeFlashColorOn Property (Page 436)
ForeFlashColorOff Property (Page 435)
FontUnderline Property (Page 434)
FontSize Property (Page 433)
FontName Property (Page 433)
FontItalic Property (Page 432)
FontBold Property (Page 432)
FlashRateForeColor Property (Page 428)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashForeColor Property (Page 424)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BoxCount Property (Page 357)
BoxAlignment Property (Page 356)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderBackColor Property (Page 353)
BackFlashColorOff Property (Page 324)
BackColor Property (Page 322)
AlignmentTop Property (Page 310)
AlignmentLeft Property (Page 309)
AdaptBorder Property (Page 306)

Round Button

Description

Object Type of ScreenItem Object. Represents the graphic object "Round Button"

Type Identifier in VBS

HMISwitch

Usage

In the following example, the object with the name "RoundButton1" is moved 10 pixels to the right:

'VBS51
Dim objSwitch
Set objSwitch= ScreenItems("RoundButton1")
objSwitch.Left = objSwitch.Left + 10
Sub OnClick(ByVal Item)
    "VBS52
    Dim objScreenItem
    On Error Resume Next    'Activation of error handling
    For Each objScreenItem In ScreenItems
        If objScreenItem.Type = "HMIButton" Then
            '=== Property "Text" available only for Standard-Button
            objScreenItem.Text = "Windows"
            If 0 <> Err.Number Then
                HMIRuntime.Trace objScreenItem.ObjectName & ": no Windows-Button" & vbCrLf
                Err.Clear    'Delete error message
            End If
        '=== Property "Radius" available only for RoundButton
            objScreenItem.Radius = 10
            If 0 <> Err.Number Then
                HMIRuntime.Trace ScreenItem.ObjectName & ": no RoundButton" & vbCrLf
                Err.Clear
            End If
            End If
        Next
    On Error Goto 0    'Deactivation of error handling
End Sub

See also

PicDownUseTransColor Property (Page 535)
BorderColorTop Property (Page 354)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Toggle Property (Page 627)
Radius Property (Page 547)
Pressed Property (Page 543)
PicUpUseTransColor Property (Page 540)
PicUpTransparent Property (Page 540)
PicUpReferenced Property (Page 539)
PictureUp Property (Page 539)
PictureDown Property (Page 537)
PictureDeactivated Property (Page 537)
PicDownTransparent Property (Page 535)
PicDownReferenced Property (Page 535)
PicDeactUseTransColor Property (Page 534)
PicDeactTransparent Property (Page 534)
PicDeactReferenced-Eigenschaft (Page 534)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
Enabled Property (Page 408)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
BorderFlashColorOn Property (Page 355)
BorderFlashColorOff Property (Page 354)
BorderColor Property (Page 353)
BorderColorBottom Property (Page 354)
BorderBackColor Property (Page 353)
BackFlashColorOn Property (Page 324)
BackFlashColorOff Property (Page 324)
BackColor Property (Page 322)
Slider

Description

![Object Type of ScreenItem Object. Represents the graphic object "Slider"](image)

Type Identifier in VBS

HMISlider

Usage

In the following example, the object with the name "Slider1" is moved 10 pixels to the right:

'VBS53
Dim sldSlider
Set sldSlider = ScreenItems("Slider1")
sldSlider.Left = sldSlider.Left + 10

Notes on Error Handling

Sliders and WinCC slider controls are mapped in the object model to an "HMISlider" type. Since the objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0
Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handing. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.

Examples of error handling

```vbs
Sub OnClick(Byval Item)
'VBS194
Dim ScreenItem
' activating error handling:
On Error Resume Next
For Each ScreenItem In ScreenItems
If ScreenItem.Type = "HMISlider" Then
'=== Property "BevelColorUp" only exists for a WinCC Slider Control
ScreenItem.BevelColorUp = 1
If (Err.Number <> 0) Then
HMIRuntime.Trace(ScreenItem.ObjectName + ": no Windows-Slider" + vbCrlf)
' delete error message
Err.Clear
End If
'=== Property "BorderStyle" only exists for a Windows-Slider
ScreenItem.BorderStyle = 1
If (Err.Number <> 0) Then
HMIRuntime.Trace(ScreenItem.ObjectName + ": no WinCC Slider Control" + vbCrlf)
Err.Clear
End If
End If
Next
On Error GoTo 0 ' deactivating error handling
End Sub
```

See also

- Height Property (Page 443)
- BackColorBottom Property (Page 323)
- Activate Method (Page 712)
- Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
WindowsStyle Property (Page 700)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
SmallChange Property (Page 580)
Process Property (Page 544)
PasswordLevel Property (Page 530)
Parent Property (Page 529)
OperationReport Property (Page 525)
OperationMessage Property (Page 517)
ObjectName Property (Page 512)
Min Property (Page 506)
Max Property (Page 488)
Left Property (Page 476)
Layer Object (Page 120)
FlashRateBorderColor Property (Page 427)
FlashRateBackColor Property (Page 427)
FlashBorderColor Property (Page 424)
FlashBackColor Property (Page 423)
FillStyle Property (Page 419)
FillingIndex Property (Page 419)
Filling Property (Page 419)
FillColor Property (Page 418)
ExtendedOperation Property (Page 415)
Enabled Property (Page 408)
Direction Property (Page 405)
ColorTop Property (Page 377)
ColorBottom Property (Page 375)
ButtonColor Property (Page 357)
BorderWidth Property (Page 355)
BorderStyle Property (Page 355)
1.14.3.5 Tube objects

Polygon Tube

Description

Object Type of ScreenItem Object. Represents the "Polygon Tube" graphic object.

Type Identifier in VBS

HMITubePolyline

Usage

In the following example, the object with the name "TubePolyline1" is moved 10 pixels to the right:

'VBS24
Dim objTubePolyline
Set objTubePolyline = ScreenItems("TubePolyline1")
objTubePolyline.Left = objTubePolyline.Left + 10
T-piece

Description

Object Type of ScreenItem Object. Represents the “T-piece” graphic object.

Type Identifier in VBS

HMITubeTeeObject

Usage

In the following example, the object with the name “TubeTeeObject1” is moved 10 pixels to the right:

'VBS21
Dim objTubeTeeObject
Set objTubeTeeObject = ScreenItems("TubeTeeObject1")
objTubeTeeObject.Left = objTubeTeeObject.Left + 10

Double T-piece

Description

Object Type of ScreenItem Object. Represents the "Double T-piece" graphic object.
**Type Identifier in VBS**

HMITubeDoubleTeeObject

**Usage**

In the following example, the object with the name "TubeDoubleTeeObject1" is moved 10 pixels to the right:

```vbs
'VBS21
Dim objTubeDoubleTeeObject
Set objTubeDoubleTeeObject = ScreenItems("TubeDoubleTeeObject1")
objTubeDoubleTeeObject.Left = objTubeDoubleTeeObject.Left + 10
```

**Tube Bend**

**Description**

Object Type of ScreenItem Object. Represents the "Tube Arc" graphic object.

**Type Identifier in VBS**

HMITubeArcObject

**Usage**

In the following example, the object with the name "TubeArcObject1" is moved 10 pixels to the right:

```vbs
'VBS24
Dim objTubeArcObject
Set objTubeArcObject = ScreenItems("TubeArcObject1")
objTubeArcObject.Left = objTubeArcObject.Left + 10
```
1.14.3.6 Controls

Controls

Special features with controls

In the case of non-WinCC controls, the version-independent ProgID is returned as the type. It is possible to determine the version-dependent ProgID or "User friendly Name" from the ProgID: In the following example, "Control1" is a control embedded in the picture which already returns the version-independent ProgID as a result of the Type property.

Note
Since not every Control has a version-dependent ProgID, an error handling measure should be integrated to query the version-dependent ProgID or UserFriendlyName. If no error handling is used, the code is terminated immediately without any result when no ProgID is found.

Determine the version-dependent ProgID as follows:

'VBS153
Dim objControl
Dim strCurrentVersion
Set objControl = ScreenItems("Control1")
strCurrentVersion = CreateObject("WScript.Shell").RegRead("HKCR" & objControl.Type & "\CurVer")
MsgBox strCurrentVersion

Note
In order that example above works, a multimedia control should be inserted in the picture.

Determine the User Friendly Name as follows:

'VBS154
Dim objControl
Dim strFriendlyName
Set objControl = ScreenItems("Control1")
strFriendlyName = CreateObject("WScript.Shell").RegRead("HKCR" & objControl.Type & "\")
MsgBox strFriendlyName

Note
In order that example above works, a multimedia control should be inserted in the picture.
Restrictions of VBS for Dynamization by Controls

If Controls are to be dynamized with, the following conditions must be fulfilled:

**Methods**
The "ByRef" declaration may only be implemented as a "Variant" (ByRef xxx as Variant)
The "ByVal" declaration may only be implemented with tag types (ByVal xxx as Long)

**Properties**
The "ByRef" declaration may only be implemented as a "Variant" (ByRef xxx as Variant)
The "ByVal" declaration may only be implemented with tag types (ByVal xxx as Long)

**Events**
The "ByRef" declaration is not permitted.
The "ByVal" declaration may only be implemented as a "Variant" (ByVal xxx as Variant)

**Arrays**
If arrays are used, they must be declared with (ByRef xxx As Variant).
In order that arrays can be transferred in variants, variant tag must also be inserted as an intermediate tag according to the following scheme:

```vbs
'VBS151
Dim arrayPoints(200)
Dim vArrayCoercion 'Variant for array Coercion
' Make the VBS Array compatible with the OLE Automation
vArrayCoercion = (arrayPoints)
objTrendControl.DataXY = vArrayCoercion ' this array will occur in the control
```

Use of Controls from External Suppliers

If a non-WinCC control is used, it is possible that the properties provided by the control have the same names as the general ScreenItem properties. In such cases, the ScreenItem properties have priority. The "hidden" properties of an external control supplier can be accessed using the additional "object" property. Address the properties of an external control supplier as follows:

```vbs
Control.object.type
```

If you use the following form, the properties of the ScreenItem object are used in the case of identical names:

```vbs
Control.type
```
Double parameter
When using a Control which is not an internal WinCC control, it is possible that the event prototypes contain a parameter with the name "Item". In this case, the name of the parameter is renamed according to "ObjectItem" in the VBS prototype submitted. If this name already exists, the name is differentiated by numbers being appended.

WinCC controls available
- HMI Symbol Library
- WinCC AlarmControl
- WinCC Alarm Control (before WinCC V7)
- WinCC Digital/Analog Clock
- WinCC FunctionTrendControl
- WinCC Function Trend Control (before WinCC V7)
- WinCC Gauge Control
- WinCC Media Control
- WinCC OnlineTableControl
- WinCC Online Table Control (before WinCC V7)
- WinCC OnlineTrendControl
- WinCC Online Trend Control (before WinCC V7)
- WinCC Push Button Control
- WinCC RulerControl
- WinCC Slider Control
- WinCC UserArchiveControl

See also
HMI Symbol Library (Page 239)
WinCC Slider Control (Page 277)
WinCC Push Button Control (Page 269)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Gauge Control (Page 254)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Digital/Analog Clock (Page 246)
WinCC Alarm Control (before WinCC V7) (Page 286)
WinCC UserArchiveControl (Page 280)
WinCC RulerControl (Page 272)
List of controls

Column object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Column" listing object to configure the properties of the columns in the WinCC UserArchiveControl.

Use in the controls

- WinCC UserArchiveControl (Page 280)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "colobj.ColumnName", the listing name "Column" is dropped: "colobj.Name".

<table>
<thead>
<tr>
<th>Property (Page)</th>
<th>Property (Page)</th>
<th>Property (Page)</th>
<th>Property (Page)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnAlias</td>
<td>ColumnFlagUnique</td>
<td>ColumnPosition</td>
<td>ColumnSort</td>
</tr>
<tr>
<td>(Page 379)</td>
<td>(Page 381)</td>
<td>(Page 384)</td>
<td>(Page 387)</td>
</tr>
<tr>
<td>ColumnAlign</td>
<td>ColumnHideText</td>
<td>ColumnPrecisions</td>
<td>ColumnSortIndex</td>
</tr>
<tr>
<td>(Page 379)</td>
<td>(Page 381)</td>
<td>(Page 384)</td>
<td>(Page 387)</td>
</tr>
<tr>
<td>ColumnAutoPrecisions</td>
<td>ColumnHideTitleText</td>
<td>ColumnReadAccess</td>
<td>ColumnStartValue</td>
</tr>
<tr>
<td>(Page 379)</td>
<td>(Page 382)</td>
<td>(Page 384)</td>
<td>(Page 387)</td>
</tr>
<tr>
<td>ColumnCaption</td>
<td>ColumnIndex</td>
<td>ColumnReadonly</td>
<td>ColumnStringLength</td>
</tr>
<tr>
<td>(Page 380)</td>
<td>(Page 382)</td>
<td>(Page 384)</td>
<td>(Page 387)</td>
</tr>
<tr>
<td>ColumnCount</td>
<td>ColumnLeadingZeros</td>
<td>ColumnRepos</td>
<td>ColumnTimeFormat</td>
</tr>
<tr>
<td>(Page 380)</td>
<td>(Page 383)</td>
<td>(Page 385)</td>
<td>(Page 388)</td>
</tr>
<tr>
<td>ColumnDateFormat</td>
<td>ColumnLength</td>
<td>ColumnShowDate</td>
<td>ColumnType</td>
</tr>
<tr>
<td>(Page 380)</td>
<td>(Page 383)</td>
<td>(Page 386)</td>
<td>(Page 389)</td>
</tr>
<tr>
<td>ColumnDMVarName</td>
<td>ColumnMaxValue</td>
<td>ColumnShowIcon</td>
<td>ColumnVisible</td>
</tr>
<tr>
<td>(Page 380)</td>
<td>(Page 383)</td>
<td>(Page 386)</td>
<td>(Page 389)</td>
</tr>
<tr>
<td>ColumnExponentialFormat</td>
<td>ColumnMinValue</td>
<td>ColumnShowTitleIcon</td>
<td>ColumnWriteAccess</td>
</tr>
<tr>
<td>(Page 381)</td>
<td>(Page 383)</td>
<td>(Page 386)</td>
<td>(Page 389)</td>
</tr>
<tr>
<td>ColumnFlagNotNull</td>
<td>ColumnName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Page 381)</td>
<td>(Page 383)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
See also

GetColumn method (Page 720)
GetColumnCollection method (Page 721)

HitlistColumn object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type
(users can change the number).

Use the "HitlistColumn" listing object to configure the message blocks used in the hitlist of
WinCC AlarmControl.

Use in the controls

● WinCC AlarmControl (Page 241)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the
listing. For example, when using "hitlistobj.HitlistColumnName", the listing name
"HitlistColumn" is dropped: "hitlistobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods.
These properties are characterized by the respective names, e.g. "Add", "Remove" or
"Rename".

<table>
<thead>
<tr>
<th>HitlistColumnAdd (Page 445)</th>
<th>HitlistColumnRepos (Page 446)</th>
<th>HitListMaxSourceItems (Page 448)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HitlistColumnAutoAdjust (Page 445)</td>
<td>HitlistColumnSort (Page 447)</td>
<td>HitListMaxSourceItemsWarn (Page 448)</td>
</tr>
<tr>
<td>HitlistColumnCount (Page 445)</td>
<td>HitlistColumnSortIndex (Page 447)</td>
<td>HitListRelTime (Page 448)</td>
</tr>
<tr>
<td>HitlistColumnIndex (Page 446)</td>
<td>HitlistColumnVisible (Page 447)</td>
<td>HitListRelTimeFactor (Page 449)</td>
</tr>
<tr>
<td>HitlistColumnName (Page 446)</td>
<td>HitListDefaultSort (Page 447)</td>
<td>HitListRelTimeFactorType (Page 449)</td>
</tr>
<tr>
<td>HitlistColumnRemove (Page 446)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also

GetHitlistColumn method (Page 722)
GetHistlistColumnCollection method (Page 723)
**MessageBlock object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "MessageBlock" listing object to configure the message blocks in WinCC AlarmControl.

**Use in the controls**

- WinCC AlarmControl (Page 241)

**Available properties of the object**

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "messageobj.MessageBlockName", the listing name "MessageBlock" is dropped: "messageobj.Name".

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBlockAlign</td>
<td>495</td>
</tr>
<tr>
<td>MessageBlockAutoPrecisions</td>
<td>495</td>
</tr>
<tr>
<td>MessageBlockCaption</td>
<td>496</td>
</tr>
<tr>
<td>MessageBlockCount</td>
<td>496</td>
</tr>
<tr>
<td>MessageBlockDateFormat</td>
<td>497</td>
</tr>
<tr>
<td>MessageBlockExponentialFormat</td>
<td>497</td>
</tr>
<tr>
<td>MessageBlockFlashOn</td>
<td>497</td>
</tr>
<tr>
<td>MessageBlockHideText</td>
<td>498</td>
</tr>
<tr>
<td>MessageBlockHideTitleText</td>
<td>498</td>
</tr>
<tr>
<td>MessageBlockIndex</td>
<td>499</td>
</tr>
<tr>
<td>MessageBlockLeadingZeros</td>
<td>499</td>
</tr>
<tr>
<td>MessageBlockLength</td>
<td>500</td>
</tr>
<tr>
<td>MessageBlockName</td>
<td>500</td>
</tr>
<tr>
<td>MessageBlockPrecisions</td>
<td>500</td>
</tr>
<tr>
<td>MessageBlockSelected</td>
<td>500</td>
</tr>
<tr>
<td>MessageBlockShowDate</td>
<td>501</td>
</tr>
<tr>
<td>MessageBlockShowIcon</td>
<td>501</td>
</tr>
<tr>
<td>MessageBlockShowTitleIcon</td>
<td>501</td>
</tr>
<tr>
<td>MessageBlockShowIcon</td>
<td>501</td>
</tr>
<tr>
<td>MessageBlockTextId</td>
<td>501</td>
</tr>
<tr>
<td>MessageBlockType</td>
<td>502</td>
</tr>
<tr>
<td>MessageBlockTimeFormat</td>
<td>502</td>
</tr>
</tbody>
</table>

**See also**

- GetMessageMessageBlock method (Page 724)
- GetMessageMessageBlockCollection method (Page 725)

**MessageColumn object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "MessageColumn" listing object to configure the message blocks used in the message lists of WinCC AlarmControl.
Use in the controls

- WinCC AlarmControl (Page 241)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "messagecol.MessageColumnName", the listing name "MessageColumn" is dropped: "messagecol.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>MessageColumnAdd (Page 503)</th>
<th>MessageColumnName (Page 504)</th>
<th>MessageColumnSort (Page 505)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageColumnAutoAdjust (Page 503)</td>
<td>MessageColumnRemove (Page 504)</td>
<td>MessageColumnSortIndex (Page 505)</td>
</tr>
<tr>
<td>MessageColumnCount (Page 504)</td>
<td>MessageColumnRepos (Page 505)</td>
<td>MessageColumnVisible (Page 505)</td>
</tr>
<tr>
<td>MessageColumnIndex (Page 504)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also

GetMessageColumn method (Page 727)
GetMessageColumnCollection method (Page 728)

OperatorMessage object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "OperatorMessage" listing object to configure the operator messages displayed in WinCC AlarmControl.

Use in the controls

- WinCC AlarmControl (Page 241)
Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "opmessobj.OperatorMessageName", the listing name "OperatorMessage" is dropped: "opmessobj.Name".

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OperatorMessageID</td>
<td>(518)</td>
<td>OperatorMessageSource5</td>
<td>(520)</td>
<td>OperatorMessageSourceType3</td>
<td>(523)</td>
</tr>
<tr>
<td>OperatorMessageIndex</td>
<td>(518)</td>
<td>OperatorMessageSource6</td>
<td>(521)</td>
<td>OperatorMessageSourceType4</td>
<td>(523)</td>
</tr>
<tr>
<td>OperatorMessageName</td>
<td>(518)</td>
<td>OperatorMessageSource7</td>
<td>(521)</td>
<td>OperatorMessageSourceType5</td>
<td>(523)</td>
</tr>
<tr>
<td>OperatorMessageNumber</td>
<td>(519)</td>
<td>OperatorMessageSource8</td>
<td>(521)</td>
<td>OperatorMessageSourceType6</td>
<td>(524)</td>
</tr>
<tr>
<td>OperatorMessageSelected</td>
<td>(519)</td>
<td>OperatorMessageSource9</td>
<td>(522)</td>
<td>OperatorMessageSourceType7</td>
<td>(524)</td>
</tr>
<tr>
<td>OperatorMessageSource1</td>
<td>(519)</td>
<td>OperatorMessageSource10</td>
<td>(522)</td>
<td>OperatorMessageSourceType8</td>
<td>(524)</td>
</tr>
<tr>
<td>OperatorMessageSource2</td>
<td>(519)</td>
<td>OperatorMessageSourceType1</td>
<td>(522)</td>
<td>OperatorMessageSourceType9</td>
<td>(525)</td>
</tr>
<tr>
<td>OperatorMessageSource3</td>
<td>(520)</td>
<td>OperatorMessageSourceType2</td>
<td>(522)</td>
<td>OperatorMessageSourceType10</td>
<td>(525)</td>
</tr>
<tr>
<td>OperatorMessageSource4</td>
<td>(520)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also

GetOperatorMessage method (Page 729)
GetOperatorMessageCollection method (Page 730)

Row object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Row" listing object to access the rows of the table-based controls. The Row object refers to the runtime data in the tables.

Use in the controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Page</th>
<th>Control</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCC AlarmControl</td>
<td>(241)</td>
<td>WinCC OnlineTableControl</td>
<td>(257)</td>
</tr>
<tr>
<td>WinCC RulerControl</td>
<td>(272)</td>
<td>WinCC UserArchiveControl</td>
<td>(280)</td>
</tr>
</tbody>
</table>
Available methods of the object

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectAll</td>
<td>797</td>
</tr>
<tr>
<td>UnselectAll</td>
<td>812</td>
</tr>
<tr>
<td>SelectRow</td>
<td>798</td>
</tr>
<tr>
<td>UnselectRow</td>
<td>813</td>
</tr>
</tbody>
</table>

Available properties of the object

- Count (Page 394)

Example

```vbs
'VBS371
Dim mycontrol
Set mycontrol = ScreenItems.Item("Control1")
Dim rowobj
Set rowobj = mycontrol.GetRowCollection

'Reading the property Count of the collection
HMIRuntime.Trace "RowCount: " & rowobj.Count & vbNewline

'Reading the properties CellText, CellCount an RowNumber for the items of the collection
For lIndex = 1 To rowobj.Count
  'In this example the CellText of the third column is returned
  HMIRuntime.Trace "RowCellText: " & lIndex & ": " & rowobj.item(lIndex).CellText(3) & vbNewline
  HMIRuntime.Trace "RowCellCount: " & lIndex & ": " & rowobj.item(lIndex).CellCount & vbNewline
  HMIRuntime.Trace "RowNumber: " & rowobj.item(lIndex).RowNumber & vbNewline
Next
```

See also

- GetRow method (Page 731)
- GetRowCollection method (Page 732)
- GetSelectedRow method (Page 738)
- GetSelectedRows method (Page 739)
- CellCount property (Page 364)
- CellText property (Page 366)
- RowNumber property (Page 553)

RulerBlock object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).
Use the “Block” listing object to configure the blocks of WinCC RulerControl.

Use in the controls

- WinCC RulerControl (Page 272)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "rulerblockobj.BlockName", the listing name "Block" is dropped: "rulerblockobj.Name".

<table>
<thead>
<tr>
<th>BlockAlign (Page 347)</th>
<th>BlockHideText (Page 349)</th>
<th>BlockPrecisions (Page 351)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockAutoPrevisions (Page 348)</td>
<td>BlockHideTitleText (Page 349)</td>
<td>BlockShowDate (Page 351)</td>
</tr>
<tr>
<td>BlockCaption (Page 348)</td>
<td>BlockID</td>
<td>BlockShowIcon (Page 351)</td>
</tr>
<tr>
<td>BlockCount (Page 348)</td>
<td>BlockIndex (Page 350)</td>
<td>BlockShowTitleIcon (Page 352)</td>
</tr>
<tr>
<td>BlockDateFormat (Page 348)</td>
<td>BlockLength (Page 351)</td>
<td>BlockTimeFormat (Page 352)</td>
</tr>
<tr>
<td>BlockExponentialFormat (Page 349)</td>
<td>BlockName (Page 351)</td>
<td>BlockUseSourceFormat (Page 352)</td>
</tr>
</tbody>
</table>

See also

- GetRulerBlock method (Page 733)
- GetRulerBlockCollection method (Page 734)

RulerColumn object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Column" listing object to configure the columns of the ruler window in WinCC RulerControl.

Use in the controls

- WinCC RulerControl (Page 272)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "rulercolobj.ColumnName", the listing name "Column" is dropped: "rulercolobj.Name".
Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>ColumnAdd (Page 379)</th>
<th>ColumnName (Page 383)</th>
<th>ColumnSort (Page 387)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnCount (Page 380)</td>
<td>ColumnRemove (Page 384)</td>
<td>ColumnSortIndex (Page 387)</td>
</tr>
<tr>
<td>ColumnIndex (Page 382)</td>
<td>ColumnRepos (Page 385)</td>
<td>ColumnVisible (Page 389)</td>
</tr>
</tbody>
</table>

See also

GetRulerColumn method (Page 735)
GetRulerColumnCollection method (Page 736)

StatisticAreaColumn object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Column" listing object to configure the columns of the statistic area window in WinCC RulerControl.

Use in the controls

- WinCC RulerControl (Page 272)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "statareacolobj.ColumnName", the listing name "Column" is dropped: "statareacolobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>ColumnAdd (Page 379)</th>
<th>ColumnName (Page 383)</th>
<th>ColumnSort (Page 387)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnCount (Page 380)</td>
<td>ColumnRemove (Page 384)</td>
<td>ColumnSortIndex (Page 387)</td>
</tr>
<tr>
<td>ColumnIndex (Page 382)</td>
<td>ColumnRepos (Page 385)</td>
<td>ColumnVisible (Page 389)</td>
</tr>
</tbody>
</table>

See also

GetStatisticAreaColumn method (Page 741)
GetStatisticAreaColumnCollection method (Page 742)
**StatisticResultColumn object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Column" listing object to configure the columns of the statistic window in WinCC RulerControl.

**Use in the controls**

- WinCC RulerControl (Page 272)

**Available properties of the object**

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "statrescolobj.ColumnName", the listing name "Column" is dropped: "statrescolobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>ColumnAdd (Page 379)</th>
<th>ColumnName (Page 383)</th>
<th>ColumnSort (Page 387)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnCount (Page 380)</td>
<td>ColumnRemove (Page 384)</td>
<td>ColumnSortIndex (Page 387)</td>
</tr>
<tr>
<td>ColumnIndex (Page 382)</td>
<td>ColumnRepos (Page 385)</td>
<td>ColumnVisible (Page 389)</td>
</tr>
</tbody>
</table>

**See also**

GetStatisticResultColumn method (Page 743)

GetStatisticResultColumnCollection method (Page 744)

**StatusBarElement object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "StatusBarElement" listing object to configure the properties of the statusbar of the controls.
Use in the controls

<table>
<thead>
<tr>
<th>WinCC AlarmControl (Page 241)</th>
<th>WinCC FunctionTrendControl (Page 248)</th>
<th>WinCC OnlineTableControl (Page 257)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCC OnlineTrendControl (Page 262)</td>
<td>WinCC RulerControl (Page 272)</td>
<td>WinCC UserArchiveControl (Page 280)</td>
</tr>
</tbody>
</table>

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "statusbarobj.StatusbarElementName", the listing name "StatusbarElement" is dropped: "statusbarobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>StatusbarElementAdd (Page 589)</th>
<th>StatusbarElementIndex (Page 590)</th>
<th>StatusbarElementText (Page 591)</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusbarElementAutoSize (Page 589)</td>
<td>StatusbarElementName (Page 590)</td>
<td>StatusbarElementTooltipText (Page 591)</td>
</tr>
<tr>
<td>StatusbarElementCount (Page 589)</td>
<td>StatusbarElementRemove (Page 590)</td>
<td>StatusbarElementUserDefined (Page 592)</td>
</tr>
<tr>
<td>StatusbarElementIconId (Page 589)</td>
<td>StatusbarElementRename (Page 590)</td>
<td>StatusbarElementVisible (Page 591)</td>
</tr>
<tr>
<td>StatusbarElementId (Page 589)</td>
<td>StatusbarElementRepos (Page 591)</td>
<td>StatusbarElementWidth (Page 592)</td>
</tr>
</tbody>
</table>

See also

GetStatusbarElement method (Page 745)
GetStatusbarElementCollection method (Page 746)

TimeAxis object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "TimeAxis" listing object to configure the properties of the time axis in columns in the WinCC OnlineTrendControl.

Use in the controls

- WinCC OnlineTrendControl (Page 262)
Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "timeaxisobj.TimeAxisName", the listing name "TimeAxis" is dropped: "timeaxisobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>TimeAxisActualize</th>
<th>TimeAxisIndex (Page 605)</th>
<th>TimeAxisRepos (Page 607)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeAxisAdd (Page 603)</td>
<td>TimeAxisInTrendColor</td>
<td>TimeAxisShowDate (Page 608)</td>
</tr>
<tr>
<td>TimeAxisAlign (Page 603)</td>
<td>TimeAxisLabel (Page 606)</td>
<td>TimeAxisTimeFormat (Page 608)</td>
</tr>
<tr>
<td>TimeAxisBeginTime (Page 603)</td>
<td>TimeAxisMeasurePoints (Page 606)</td>
<td>TimeAxisTimeRangeBase (Page 608)</td>
</tr>
<tr>
<td>TimeAxisColor (Page 604)</td>
<td>TimeAxisName (Page 606)</td>
<td>TimeAxisTimeRangeFactor (Page 609)</td>
</tr>
<tr>
<td>TimeAxisCount (Page 604)</td>
<td>TimeAxisRangeType (Page 607)</td>
<td>TimeAxisTrendWindow (Page 609)</td>
</tr>
<tr>
<td>TimeAxisDateFormat (Page 604)</td>
<td>TimeAxisRemove (Page 607)</td>
<td>TimeAxisVisible (Page 609)</td>
</tr>
<tr>
<td>TimeAxisEndTime (Page 604)</td>
<td>TimeAxisRename (Page 607)</td>
<td></td>
</tr>
</tbody>
</table>

See also

GetTimeAxis method (Page 747)
GetTimeAxisCollection method (Page 748)

TimeColumn object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "TimeColumn" listing object to configure the properties of the time column in WinCC OnlineTrendControl.

Use in the controls

- WinCC OnlineTableControl (Page 257)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "timecolobj.TimeColumnName", the listing name "TimeColumn" is dropped: "timecolobj.Name".
Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

| TimeColumnActualize (Page 610) | TimeColumnHideText (Page 613) | TimeColumnShowDate (Page 616) |
| TimeColumnAdd (Page 610) | TimeColumnHideTitleText (Page 614) | TimeColumnShowIcon (Page 616) |
| TimeColumnAlign (Page 611) | TimeColumnIndex (Page 614) | TimeColumnShowTitleIcon (Page 616) |
| TimeColumnBackColor (Page 611) | TimeColumnLength (Page 614) | TimeColumnSort (Page 617) |
| TimeColumnBeginTime (Page 612) | TimeColumnMeasurePoints (Page 614) | TimeColumnSortIndex (Page 617) |
| TimeColumnCaption (Page 612) | TimeColumnName (Page 615) | TimeColumnBackColor (Page 611) |
| TimeColumnCount (Page 612) | TimeColumnRangeType (Page 615) | TimeColumnTimeFormat (Page 617) |
| TimeColumnDateFormat (Page 612) | TimeColumnRemove (Page 615) | TimeColumnTimeRangeBase (Page 618) |
| TimeColumnEndTime (Page 613) | TimeColumnRename (Page 615) | TimeColumnTimeRangeFactor (Page 618) |
| TimeColumnForeColor (Page 613) | TimeColumnRepos (Page 616) | TimeColumnVisible (Page 618) |

See also

- GetTimeColumn method (Page 750)
- GetTimeColumnCollection method (Page 751)

**ToolbarButton object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "ToolbarButton" listing object to configure the properties of the toolbar of the controls.

**Use in the controls**

- WinCC AlarmControl (Page 241)
- WinCC FunctionTrendControl (Page 248)
- WinCC OnlineTableControl (Page 257)
- WinCC OnlineTrendControl (Page 262)
- WinCC RulerControl (Page 272)
- WinCC UserArchiveControl (Page 280)

**Available properties of the object**

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "toolbarobj.ToolbarButtonName", the listing name "ToolbarButton" is dropped: "toolbarobj.Name".
Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>ToolbarButtonActive (Page 630)</th>
<th>ToolbarButtonId (Page 635)</th>
<th>ToolbarButtonRename (Page 637)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarButtonAdd (Page 630)</td>
<td>ToolbarButtonIndex (Page 636)</td>
<td>ToolbarButtonRepos (Page 637)</td>
</tr>
<tr>
<td>ToolbarButtonBeginGroup (Page 630)</td>
<td>ToolbarButtonLocked (Page 636)</td>
<td>ToolbarButtonTooltipText (Page 638)</td>
</tr>
<tr>
<td>ToolbarButtonCount (Page 635)</td>
<td>ToolbarButtonName (Page 636)</td>
<td>ToolbarButtonUserDefined (Page 638)</td>
</tr>
<tr>
<td>ToolbarButtonEnabled (Page 635)</td>
<td>ToolbarButtonPasswordLevel (Page 636)</td>
<td>ToolbarButtonVisible (Page 638)</td>
</tr>
<tr>
<td>ToolbarButtonHotKey (Page 635)</td>
<td>ToolbarButtonRemove (Page 636)</td>
<td></td>
</tr>
</tbody>
</table>

See also
- GetToolbarButton method (Page 752)
- GetToolbarButtonCollection method (Page 753)

Trend object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Trend" listing object to configure the properties of the trends. The "InsertData" and "RemoveData" methods are used to fill the trend with data or to delete the trend. The "GetRulerData" method is used to access the data at a particular point of the trend.

Use in the controls

- WinCC FunctionTrendControl (Page 248)
- WinCC OnlineTrendControl (Page 262)

Available methods of the object

- GetRulerData (Page 737)
- InsertData (Page 769)
- RemoveData (Page 794)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "trendobj.Trendname", the listing name "Trend" is dropped: "trendobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".
### Properties in WinCC FunctionTrendControl and WinCC OnlineTrendControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrendAdd</td>
<td>645</td>
<td>TrendLineWidth</td>
<td>650</td>
<td>TrendRemove</td>
<td>654</td>
</tr>
<tr>
<td>TrendColor</td>
<td>647</td>
<td>TrendLowerLimit</td>
<td>650</td>
<td>TrendRename</td>
<td>654</td>
</tr>
<tr>
<td>TrendCount</td>
<td>647</td>
<td>TrendLowerLimitColor</td>
<td>650</td>
<td>TrendRepos</td>
<td>654</td>
</tr>
<tr>
<td>TrendExtendedColorSet</td>
<td></td>
<td>TrendLowerLimitColoring</td>
<td>651</td>
<td>TrendTrendWindow</td>
<td>657</td>
</tr>
<tr>
<td>TrendFill</td>
<td>648</td>
<td>TrendName</td>
<td>651</td>
<td>TrendUncertainColor</td>
<td>657</td>
</tr>
<tr>
<td>TrendFillColor</td>
<td>648</td>
<td>TrendPointColor</td>
<td>651</td>
<td>TrendUncertainColoring</td>
<td>658</td>
</tr>
<tr>
<td>TrendFillColor</td>
<td>648</td>
<td>TrendPointStyle</td>
<td></td>
<td>TrendUncertainColoring</td>
<td>658</td>
</tr>
<tr>
<td>TrendIndex</td>
<td>649</td>
<td>TrendPointColor</td>
<td>651</td>
<td>TrendUnsureColor</td>
<td>658</td>
</tr>
<tr>
<td>TrendIndex</td>
<td>649</td>
<td>TrendProviderCLSID</td>
<td>653</td>
<td>TrendVisible</td>
<td>660</td>
</tr>
<tr>
<td>TrendLabel</td>
<td>649</td>
<td>TrendProvider</td>
<td>652</td>
<td>TrendUpperLimit</td>
<td>658</td>
</tr>
<tr>
<td>TrendLabel</td>
<td>649</td>
<td>TrendProviderCLSID</td>
<td>653</td>
<td>TrendUpperLimitColoring</td>
<td>658</td>
</tr>
<tr>
<td>TrendLineEdit</td>
<td>649</td>
<td>TrendProviderCLSID</td>
<td>653</td>
<td>TrendTrendWindow</td>
<td>657</td>
</tr>
<tr>
<td>TrendLineType</td>
<td>650</td>
<td>TrendProviderCLSID</td>
<td>653</td>
<td>TrendVisible</td>
<td>660</td>
</tr>
<tr>
<td>TrendRangeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Properties in WinCC OnlineTrendControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrendAutoRangeBeginTagName</td>
<td>645</td>
<td>TrendAutoRangeSource</td>
<td>646</td>
<td>TrendValueAlignment</td>
<td></td>
</tr>
<tr>
<td>TrendAutoRangeBeginValue</td>
<td>646</td>
<td>TrendSelectTagName</td>
<td>654</td>
<td>TrendValueAxis</td>
<td>659</td>
</tr>
<tr>
<td>TrendAutoRangeEndTagName</td>
<td>646</td>
<td>TrendTagName</td>
<td>656</td>
<td>TrendValueUnit</td>
<td></td>
</tr>
<tr>
<td>TrendAutoRangeEndValue</td>
<td>646</td>
<td>TrendTimeAxis</td>
<td>656</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Properties in the WinCC FunctionTrendControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrendActualize</td>
<td>645</td>
<td>TrendSelectTagNameX</td>
<td>655</td>
<td>TrendTimeRangeBase</td>
<td>656</td>
</tr>
<tr>
<td>TrendBeginTime</td>
<td>647</td>
<td>TrendSelectTagNameY</td>
<td>655</td>
<td>TrendTimeRangeFactor</td>
<td>657</td>
</tr>
<tr>
<td>TrendEndTime</td>
<td>648</td>
<td>TrendTagNameX</td>
<td>656</td>
<td>TrendXAxis</td>
<td>666</td>
</tr>
<tr>
<td>TrendMeasurePoints</td>
<td>651</td>
<td>TrendTagNameY</td>
<td>656</td>
<td>TrendYAxis</td>
<td>666</td>
</tr>
<tr>
<td>TrendRangeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### See also

- GetTrend method (Page 754)
- GetTrendCollection method (Page 755)

### TrendWindow object (list)

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the “TrendWindow” listing object to configure the properties of the trend window.
Use in the controls

| WinCC FunctionTrendControl (Page 248) | WinCC OnlineTrendControl (Page 262) |

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "trendwndobj.TrendWindowName", the listing name "TrendWindow" is dropped: "trendwndobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

### Properties in WinCC FunctionTrendControl and WinCC OnlineTrendControl

<table>
<thead>
<tr>
<th>TrendWindowAdd (Page 660)</th>
<th>TrendWindowGridInTrendColor (Page 662)</th>
<th>TrendWindowRulerColor (Page 663)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrendWindowCoarseGrid (Page 660)</td>
<td>TrendWindowHorizontalGrid (Page 662)</td>
<td>TrendWindowRulerLayer</td>
</tr>
<tr>
<td>TrendWindowCoarseGridColor (Page 660)</td>
<td>TrendWindowIndex (Page 662)</td>
<td>TrendWindowRulerStyle (Page 664)</td>
</tr>
<tr>
<td>TrendWindowCount (Page 661)</td>
<td>TrendWindowName (Page 662)</td>
<td>TrendWindowRulerWidth (Page 664)</td>
</tr>
<tr>
<td>TrendWindowFineGrid (Page 661)</td>
<td>TrendWindowRemove (Page 663)</td>
<td>TrendWindowSpacePortion (Page 664)</td>
</tr>
<tr>
<td>TrendWindowFineGridColor (Page 661)</td>
<td>TrendWindowRename (Page 663)</td>
<td>TrendWindowVerticalGrid (Page 665)</td>
</tr>
<tr>
<td>TrendWindowForegroundTrendGrid (Page 661)</td>
<td>TrendWindowRepos (Page 663)</td>
<td>TrendWindowVisible (Page 666)</td>
</tr>
</tbody>
</table>

### Properties in WinCC OnlineTrendControl

| TrendWindowStatisticRulerColor (Page 665) | TrendWindowStatisticRulerStyle (Page 665) | TrendWindowStatisticRulerWidth (Page 665) |

### See also

GetTrendWindow method (Page 757)
GetTrendWindowCollection method (Page 758)

### ValueAxis object (list)

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "ValueAxis" listing object to configure the properties of the value axis in WinCC OnlineTrendControl.
Use in the controls

- WinCC OnlineTrendControl (Page 262)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "valueaxisobj.ValueAxisName", the listing name "ValueAxis" is dropped: "valueaxisobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>ValueAxisAdd (Page 681)</th>
<th>ValueAxisEndValue (Page 683)</th>
<th>ValueAxisRemove (Page 686)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueAxisAlign (Page 681)</td>
<td>ValueAxisExponentialFormat (Page 684)</td>
<td>ValueAxisRename (Page 686)</td>
</tr>
<tr>
<td>ValueAxisAutoPrecisions (Page 682)</td>
<td>ValueAxisIndex (Page 684)</td>
<td>ValueAxisRepos (Page 686)</td>
</tr>
<tr>
<td>ValueAxisAutoRange (Page 682)</td>
<td>ValueAxisInTrendColor</td>
<td>ValueAxisScalingType (Page 687)</td>
</tr>
<tr>
<td>ValueAxisBeginValue (Page 683)</td>
<td>ValueAxisLabel (Page 685)</td>
<td>ValueAxisTrendWindow (Page 687)</td>
</tr>
<tr>
<td>ValueAxisColor (Page 683)</td>
<td>ValueAxisName (Page 685)</td>
<td>ValueAxisVisible (Page 687)</td>
</tr>
<tr>
<td>ValueAxisCount (Page 683)</td>
<td>ValueAxisPrecisions (Page 686)</td>
<td></td>
</tr>
</tbody>
</table>

See also

GetValueAxis method (Page 759)
GetValueAxisCollection method (Page 760)

ValueColumn object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "ValueColumn" listing object to configure the properties of the value column in WinCC OnlineTrendControl.

Use in the controls

- WinCC OnlineTableControl (Page 257)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "valcolobj.ValueColumnName", the listing name "ValueColumn" is dropped: "valcolobj.Name".
Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Page Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueColumnAdd</td>
<td>687</td>
</tr>
<tr>
<td>ValueColumnHideTitleText</td>
<td>690</td>
</tr>
<tr>
<td>ValueColumnRepos</td>
<td>692</td>
</tr>
<tr>
<td>ValueColumnAlign</td>
<td>688</td>
</tr>
<tr>
<td>ValueColumnIndex</td>
<td>690</td>
</tr>
<tr>
<td>ValueColumnSelectTagName</td>
<td>693</td>
</tr>
<tr>
<td>ValueColumnAutoPrecisions</td>
<td>688</td>
</tr>
<tr>
<td>ValueColumnLength</td>
<td>691</td>
</tr>
<tr>
<td>ValueColumnShowIcon</td>
<td>693</td>
</tr>
<tr>
<td>ValueColumnBackColor</td>
<td>689</td>
</tr>
<tr>
<td>ValueColumnName</td>
<td>691</td>
</tr>
<tr>
<td>ValueColumnShowTitleIcon</td>
<td>693</td>
</tr>
<tr>
<td>ValueColumnCaption</td>
<td>689</td>
</tr>
<tr>
<td>ValueColumnPrecisions</td>
<td>691</td>
</tr>
<tr>
<td>ValueColumnSort</td>
<td>693</td>
</tr>
<tr>
<td>ValueColumnCount</td>
<td>689</td>
</tr>
<tr>
<td>ValueColumnProvider</td>
<td>691</td>
</tr>
<tr>
<td>ValueColumnSortIndex</td>
<td>694</td>
</tr>
<tr>
<td>ValueColumnExponentialFormat</td>
<td>689</td>
</tr>
<tr>
<td>ValueColumnProviderCLSID</td>
<td>692</td>
</tr>
<tr>
<td>ValueColumnTagName</td>
<td>694</td>
</tr>
<tr>
<td>ValueColumnForeColor</td>
<td>690</td>
</tr>
<tr>
<td>ValueColumnRemove</td>
<td>692</td>
</tr>
<tr>
<td>ValueColumnTimeColumn</td>
<td>694</td>
</tr>
<tr>
<td>ValueColumnHideText</td>
<td>690</td>
</tr>
<tr>
<td>ValueColumnRename</td>
<td>692</td>
</tr>
<tr>
<td>ValueColumnVisible</td>
<td>695</td>
</tr>
</tbody>
</table>

**See also**

GetValueColumn method (Page 761)
GetValueColumnCollection method (Page 762)

**XAxis object (list)**

**Description**

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Xaxis" listing object to configure the properties of the X axis in WinCC FunctionTrendControl.

**Use in the controls**

- WinCC FunctionTrendControl (Page 248)

**Available properties of the object**

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "xaxisobj.XAxisName", the listing name "XAxis" is dropped: "xaxisobj.Name".
Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>XAxisAdd (Page 702)</th>
<th>XAxisEndValue (Page 704)</th>
<th>XAxisRemove (Page 706)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAxisAlign</td>
<td>XAxisExponentialFormat (Page 704)</td>
<td>XAxisRename (Page 708)</td>
</tr>
<tr>
<td>XAxisAutoPrecisions (Page 703)</td>
<td>XAxisIndex (Page 708)</td>
<td>XAxisRepos (Page 706)</td>
</tr>
<tr>
<td>XAxisAutoRange (Page 703)</td>
<td>XAxisInTrendColor</td>
<td>XAxisScalingType (Page 706)</td>
</tr>
<tr>
<td>XAxisBeginValue (Page 703)</td>
<td>XAxisLabel (Page 705)</td>
<td>XAxisTrendWindow (Page 707)</td>
</tr>
<tr>
<td>XAxisColor (Page 704)</td>
<td>XAxisName (Page 705)</td>
<td>XAxisVisible (Page 707)</td>
</tr>
<tr>
<td>XAxisCount (Page 707)</td>
<td>XAxisPrecisions (Page 706)</td>
<td></td>
</tr>
</tbody>
</table>

See also

GetXAxis method (Page 764)
GetXAxisCollection method (Page 765)

YAxis object (list)

Description

The listing of controls is a data container that can save a number of objects of the same type (users can change the number).

Use the "Yaxis" listing object to configure the properties of the Y axis in WinCC FunctionTrendControl.

Use in the controls

- WinCC FunctionTrendControl (Page 248)

Available properties of the object

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "yaxisobj.YAxisName", the listing name "YAxis" is dropped: "yaxisobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>YAxisAdd (Page 702)</th>
<th>YAxisEndValue (Page 704)</th>
<th>YAxisRemove (Page 706)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAxisAlign</td>
<td>YAxisExponentialFormat (Page 704)</td>
<td>YAxisRename (Page 709)</td>
</tr>
<tr>
<td>YAxisAutoPrecisions (Page 703)</td>
<td>YAxisIndex (Page 708)</td>
<td>YAxisRepos (Page 706)</td>
</tr>
<tr>
<td>YAxisAutoRange (Page 703)</td>
<td>YAxisInTrendColor</td>
<td>YAxisScalingType (Page 706)</td>
</tr>
<tr>
<td>YAxisBeginValue (Page 703)</td>
<td>YAxisLabel (Page 705)</td>
<td>YAxisTrendWindow (Page 707)</td>
</tr>
</tbody>
</table>
See also

GetYAxis method (Page 766)
GetYAxisCollection method (Page 767)

HMI Symbol Library

Description

Object Type of ScreenItem Object. Represents the graphic object "HMI Symbol Library"

Type Identifier in VBS

HMISymbolLibrary

Usage

In the following example, the object with the name "Control1" is moved 20 pixels to the right:

'VBS64
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +20

Properties

This object type has the following properties:
See also

Left Property (Page 476)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Controls (Page 218)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
Stretch Property (Page 594)
Picture Property (Page 536)
Parent Property (Page 529)
ObjectName Property (Page 512)
Object Property (Page 512)
Layer Object (Page 120)
Height Property (Page 443)
ForeColor Property (Page 435)
Flip Property (Page 429)
Enabled Property (Page 408)
Cursor Property (Page 395)
BlinkColor Property (Page 346)
BackStyle Property (Page 325)
BackColor Property (Page 322)
WinCC AlarmControl

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC AlarmControl" as of WinCC V7.0.

Type Identifier in VBS

HMIAlarmControl

Available list objects

<table>
<thead>
<tr>
<th>HitlistColumn (Page 222)</th>
<th>Row (Page 225)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBlock (Page 223)</td>
<td>StatusbarElement (Page 229)</td>
</tr>
<tr>
<td>MessageColumn (Page 223)</td>
<td>ToolbarButton (Page 232)</td>
</tr>
<tr>
<td>OperatorColumn (Page 224)</td>
<td></td>
</tr>
</tbody>
</table>

Methods Available in VBS

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic</th>
<th>AttachDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CopyRows</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeactivateDynamic</td>
<td>DetachDB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GetHitlistColumn</th>
<th>GetOperatorMessage</th>
<th>GetStatusbarElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHitlistColumnCollection</td>
<td>GetOperatorMessageCollection</td>
<td>GetStatusbarElementCollection</td>
</tr>
<tr>
<td>GetMessageBlock</td>
<td>GetRow (Page 731)</td>
<td>GetToolbarButton</td>
</tr>
</tbody>
</table>

|                  |                      |
|                  |                      |
|                  |                      |
|                  |                      |
|                  |                      |
Properties Available in VBS

If you access the properties with a listing object, you do not have to enter the name of the listing. For example, when using "messagecol.MessageColumnName", the listing name "MessageColumn" is dropped: "messagecol.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".
<table>
<thead>
<tr>
<th>Property</th>
<th>Property</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackColor</td>
<td>BorderColor</td>
<td>BorderWidth</td>
</tr>
<tr>
<td>Caption</td>
<td>CellSpaceRight</td>
<td>ColumnTitleAlign</td>
</tr>
<tr>
<td>CellCut</td>
<td>CellSpaceTop</td>
<td>ColumnTitles</td>
</tr>
<tr>
<td>CellSpaceBottom</td>
<td>Closeable</td>
<td></td>
</tr>
<tr>
<td>CellSpaceLeft</td>
<td>ColumnResize</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DefaultMsgFilterSQL</td>
<td>DefaultSort2</td>
<td>DisplayOptions</td>
</tr>
<tr>
<td>DefaultSort</td>
<td>DefaultSort2Column</td>
<td>DoubleClickAction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExportDirectoryChangeable</td>
<td>ExportFilenameChangeable</td>
<td>ExportSelection</td>
</tr>
<tr>
<td>ExportDirectoryname</td>
<td>ExportFormatGuid</td>
<td>ExportShowDialog</td>
</tr>
<tr>
<td>ExportFileExtension</td>
<td>ExportFormatName</td>
<td>ExportXML</td>
</tr>
<tr>
<td>ExportFilename</td>
<td>ExportParameters</td>
<td></td>
</tr>
<tr>
<td>Font</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GridLineColor</td>
<td>GridLineWidth</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HitlistColumnAdd</td>
<td>HitlistColumnRepos</td>
<td>HitListMaxSourceItemsWarn</td>
</tr>
<tr>
<td>HitlistColumnAutoAdjust (Page 445)</td>
<td>HitlistColumnSort</td>
<td>HitListRelTime</td>
</tr>
<tr>
<td>HitlistColumnCount</td>
<td>HitlistColumnSortIndex</td>
<td>HitListRelTimeFactorType</td>
</tr>
<tr>
<td>HitlistColumnName</td>
<td>HitlistColumnVisible</td>
<td>HitListRelTimeFactorType</td>
</tr>
<tr>
<td>HitlistColumnRemove</td>
<td>HitListDefaultSort</td>
<td>HorizontalGridLines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IconSpace</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LineColor (Page 480)</td>
<td>LineWidth (Page 481)</td>
<td>LongTermArchiveConsistency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MessageBlockAlign</td>
<td>MessageBlockLength</td>
<td>MessageColumnIndex</td>
</tr>
<tr>
<td>MessageBlockAutoPrecisions</td>
<td>MessageBlockPrecisions</td>
<td>MessageColumnName</td>
</tr>
<tr>
<td>MessageBlockCaption</td>
<td>MessageBlockSelected</td>
<td>MessageColumnRemove</td>
</tr>
</tbody>
</table>
### 1.14 VBS Reference

<table>
<thead>
<tr>
<th>MessageBlockCount</th>
<th>MessageBlockShowDate</th>
<th>MessageColumnRepos</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBlockDateFormat</td>
<td>MessageBlockShowIcon</td>
<td>MessageColumnSort</td>
</tr>
<tr>
<td>MessageBlockExponentialFormat</td>
<td>MessageBlockShowTitleIcon</td>
<td>MessageColumnSortIndex</td>
</tr>
<tr>
<td>MessageBlockFlashOn</td>
<td>MessageBlockTextId</td>
<td>MessageColumnVisible</td>
</tr>
<tr>
<td>MessageBlockHideText</td>
<td>MessageColumnType</td>
<td>MessageListType</td>
</tr>
<tr>
<td>MessageBlockHideTitleText</td>
<td>MessageColumnAdd</td>
<td>Moveable</td>
</tr>
<tr>
<td>MessageBlockIndex</td>
<td>MessageColumnAutoAdjust</td>
<td>MsgFilterSQL</td>
</tr>
<tr>
<td>MessageBlockLeadingZeros</td>
<td>MessageColumnCount</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OperatorMessageID</th>
<th>OperatorMessageSource5</th>
<th>OperatorMessageSourceType4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OperatorMessageIndex</td>
<td>OperatorMessageSource6</td>
<td>OperatorMessageSourceType5</td>
</tr>
<tr>
<td>OperatorMessageName</td>
<td>OperatorMessageSource7</td>
<td>OperatorMessageSourceType6</td>
</tr>
<tr>
<td>OperatorMessageNumber</td>
<td>OperatorMessageSource8</td>
<td>OperatorMessageSourceType7</td>
</tr>
<tr>
<td>OperatorMessageSelected</td>
<td>OperatorMessageSource9</td>
<td>OperatorMessageSourceType8</td>
</tr>
<tr>
<td>OperatorMessageSource1</td>
<td>OperatorMessageSource10</td>
<td>OperatorMessageSourceType9</td>
</tr>
<tr>
<td>OperatorMessageSource2</td>
<td>OperatorMessageSourceType1</td>
<td>OperatorMessageSourceType10</td>
</tr>
<tr>
<td>OperatorMessageSource3</td>
<td>OperatorMessageSourceType2</td>
<td>OperatorMessageSourceType3</td>
</tr>
<tr>
<td>OperatorMessageSource4</td>
<td>OperatorMessageSourceType1</td>
<td>OperatorMessageSourceType3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageMode</th>
<th>PageModeMessageNumber</th>
<th>PrintJobName</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RowCellCount (Page 364)</th>
<th>RowScrollbar</th>
<th>RTPersistencePasswordLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowCellText (Page 366)</td>
<td>RowTitleAlign</td>
<td>RTPersistenceType</td>
</tr>
<tr>
<td>RowCount (Page 394)</td>
<td>RowTitles</td>
<td></td>
</tr>
<tr>
<td>RowNumber (Page 553)</td>
<td>RTPersistence</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SelectedCellColor</th>
<th>SelectionRect</th>
<th>ShowSortIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectedCellForeColor</td>
<td>SelectionRectColor</td>
<td>ShowTitle</td>
</tr>
<tr>
<td>SelectedRowColor</td>
<td>SelectionRectWidth</td>
<td>Sizeable</td>
</tr>
<tr>
<td>SelectedRowForeColor</td>
<td>SelectionType</td>
<td>SkinName</td>
</tr>
<tr>
<td>SelectedTitleColor</td>
<td>ServerNames</td>
<td>SortSequence</td>
</tr>
<tr>
<td>SelectedTitleForeColor</td>
<td>ShowSortButton</td>
<td></td>
</tr>
<tr>
<td>SelectionColoring</td>
<td>ShowSortIcon</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StatusbarBackColor</th>
<th>StatusbarElementName</th>
<th>StatusbarElementVisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusbarElementAdd</td>
<td>StatusbarElementRemove</td>
<td>StatusbarElementWidth</td>
</tr>
<tr>
<td>StatusbarElementAutoSize</td>
<td>StatusbarElementRename</td>
<td>StatusbarFontColor</td>
</tr>
</tbody>
</table>
Example

A selection of messages is defined in an existing WinCC AlarmControl. The column properties are configured in the script.

Requirement

- A "WinCC AlarmControl" with the name "Control1" has already been inserted in a process picture in Graphics Designer. The picture "C_015_Native_Alarms_Sel.pdl" from the demo project was used for this example.

- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.

- You have already configured messages in your project. Or you are using the demo project from which we have taken the messages used for the example.

- Messages have already been triggered in Runtime. The buttons "incoming" and "outgoing" were clicked in the demo project.
'VBS366
Sub OnClick(ByVal Item)
Dim objControl
Dim objMessColumn
Dim objMessBlock

Set objControl = ScreenItems("Control1")
objControl.ApplyProjectSettings = False
Set objMessBlock = objControl.GetMessageBlock("Date")
objMessBlock.DateFormat = "dd.MM.yy"
Set objMessColumn = objControl.GetMessageColumn("Time")
objMessColumn.Visible = False
objControl.MsgFilterSQL = "MSGNR >= 5 AND Priority = 0"
End Sub

Note
More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".

See also
Controls (Page 218)

WinCC Digital/Analog Clock

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Digital/Analog Clock"

Type Identifier in VBS
HMIClock
Usage

In the following example, the object with the name "Control1" is moved 11 pixels to the right:

```vbs
'VBS55
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +11
```

See also

- Parent Property (Page 529)
- Activate Method (Page 712)
- Properties (Page 301)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Controls (Page 218)
- Width Property (Page 698)
- Visible Property (Page 696)
- Type Property (Page 666)
- Top Property (Page 643)
- Ticks Property (Page 601)
- TicksColor Property (Page 601)
- SquareExtent Property (Page 587)
- SecondNeedleProperty (Page 563)
- SecondNeedleHeight Property (Page 563)
- Picture Property (Page 536)
- ObjectName Property (Page 512)
- Object Property (Page 512)
- MinuteNeedleWidth Property (Page 507)
- MinuteNeedleHeight Property (Page 507)
- LocaleID Property (Page 482)
- Left Property (Page 476)
- Layer Object (Page 120)
- HourNeedleWidth Property (Page 450)
- HourNeedleHeight Property (Page 450)
- Height Property (Page 443)
Handtype Property (Page 443)
HandFillColor Property (Page 442)
ForeColor Property (Page 435)
Font property (before WinCC V7) (Page 431)
FocusRect Property (Page 430)
Enabled Property (Page 408)
BackStyle Property (Page 325)
BackColor Property (Page 322)
Analog Property (Page 311)

WinCC FunctionTrendControl

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC FunctionTrendControl" as of WinCC V7.0.

Type Identifier in VBS
HMIFunctionTrendControl

Available list objects

<table>
<thead>
<tr>
<th>StatusbarElement (Page 229)</th>
<th>Trend (Page 233)</th>
<th>XAxis (Page 237)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarButton (Page 232)</td>
<td>TrendWindow (Page 234)</td>
<td>YAxis (Page 238)</td>
</tr>
</tbody>
</table>

Methods Available in VBS

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic</th>
<th>AttachDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeactivateDynamic</td>
<td>DetachDB</td>
<td></td>
</tr>
</tbody>
</table>
Properties Available in VBS

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "xaxisobj.XAxisName", the listing name "XAxis" is dropped: "xaxisobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".
### VBS Reference

<table>
<thead>
<tr>
<th>ExportDirectoryChangeable</th>
<th>ExportFilenameChangeable</th>
<th>ExportShowDialog</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportDirectoryName</td>
<td>ExportFormatGuid</td>
<td>ExportParameters</td>
</tr>
<tr>
<td>ExportFileExtension</td>
<td>ExportFormatName</td>
<td>ExportXML</td>
</tr>
<tr>
<td>ExportFilename</td>
<td>ExportSelection</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Font</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GraphDirection</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LineColor</th>
<th>LineWidth</th>
<th>LoadDataImmediately</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Moveable</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Online</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PrintJobName</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RTPersistence</th>
<th>RTPersistencePasswordLevel</th>
<th>RTPersistenceType</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ShowRuler</th>
<th>ShowTitle</th>
<th>SkinName</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowRuleInAxis</td>
<td>Sizeable</td>
<td></td>
</tr>
<tr>
<td>ShowScrollbars</td>
<td>ShowTrendIcon</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StatusBarBackColor</th>
<th>StatusBarElementName</th>
<th>StatusBarElementVisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusBarElementAdd</td>
<td>StatusBarElementRemove</td>
<td>StatusBarElementWidth</td>
</tr>
<tr>
<td>StatusBarElementAutoSize</td>
<td>StatusBarElementRename</td>
<td>StatusBarFontColor</td>
</tr>
<tr>
<td>StatusBarElementCount</td>
<td>StatusBarElementRepos</td>
<td>StatusBarShowToolTips</td>
</tr>
<tr>
<td>StatusBarElementIId</td>
<td>StatusBarElementText</td>
<td>StatusBarText</td>
</tr>
<tr>
<td>StatusBarElementIId</td>
<td>StatusBarElementTooltipText</td>
<td>StatusBarUseBackColor</td>
</tr>
<tr>
<td>StatusBarElementIIndex</td>
<td>StatusBarElementUserDefined</td>
<td>StatusBarVisible</td>
</tr>
</tbody>
</table>

**WinCC: Scripting (VBS, ANSI-C, VBA)**

System Manual, 02/2016, A5E37536061-AA

250
<table>
<thead>
<tr>
<th>TimeBase</th>
<th>ToolbarButtonHotKey</th>
<th>ToolbarButtonTooltipText</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarAlignment</td>
<td>ToolbarButtonId</td>
<td>ToolbarButtonUserDefined</td>
</tr>
<tr>
<td>ToolbarBackColor</td>
<td>ToolbarButtonIndex</td>
<td>ToolbarButtonVisible</td>
</tr>
<tr>
<td>ToolbarButtonActive</td>
<td>ToolbarButtonLocked</td>
<td>ToolbarShowTooltips</td>
</tr>
<tr>
<td>ToolbarButtonAdd</td>
<td>ToolbarButtonName</td>
<td>ToolbarUseBackColor</td>
</tr>
<tr>
<td>ToolbarButtonBeginGroup</td>
<td>ToolbarButtonPasswordLevel</td>
<td>ToolbarUseHotKeys</td>
</tr>
<tr>
<td>ToolbarButtonClick</td>
<td>ToolbarButtonRemove</td>
<td>ToolbarVisible</td>
</tr>
<tr>
<td>ToolbarButtonCount</td>
<td>ToolbarButtonRename</td>
<td>ToolbarUseBackColor</td>
</tr>
<tr>
<td>ToolbarButtonEnabled</td>
<td>ToolbarButtonRepos</td>
<td></td>
</tr>
</tbody>
</table>

| TrendActualize | TrendLowerLimit | TrendSelectTagNameX |
| TrendAdd | TrendLowerLimitColor | TrendSelectTagNameY |
| TrendBeginTime | TrendLowerLimitColoring | TrendTagNameX |
| TrendColor | TrendMeasurePoints | TrendTagNameY |
| TrendCount | TrendName | TrendTimeRangeBase |
| TrendEndTime | TrendPointColor | TrendTimeRangeFactor |
| TrendExtendedColorSet | TrendPointStyle | TrendTrendWindow |
| TrendFill | TrendPointWidth | TrendUncertainColor |
| TrendFillColor | TrendProvider | TrendUncertainColoring |
| TrendIndex | TrendProviderCLSID | TrendUpperLimit |
| TrendLabel | TrendRangeType | TrendUpperLimitColor |
| TrendLineStyle | TrendRemove | TrendUpperLimitColoring |
| TrendLineWidth | TrendRename | TrendVisible |
| TrendLineWidth | TrendRepos |

| TrendWindowAdd | TrendWindowHorizontalGrid | TrendWindowRulerStyle |
| TrendWindowCoarseGrid | TrendWindowIndex | TrendWindowRulerWidth |
| TrendWindowCount | TrendWindowName | TrendWindowSpacePortion |
| TrendWindowCoarseGridColor | TrendWindowRemove | TrendWindowVerticalGrid |
| TrendWindowFineGrid | TrendWindowRename | TrendWindowVisible |
| TrendWindowFineGridColor | TrendWindowRepos | TrendXAxis |
| TrendWindowForegroundTrendGrid | TrendWindowRulerColor | TrendYAxis |
| TrendWindowGridInTrendColor | TrendWindowRulerLayer |

| UseTrendNameAsLabel |

| XAxisAdd | XAxisEndValue | XAxisRemove |
| XAxisAlign | XAxisExponentialFormat | XAxisRename / YAxisRename |
### Examples

A trend is displayed in a WinCC FunctionTrendControl that is linked with a user archive. Different properties are configured for the trend in the script. The "StartID" of the user archive and the number of measurement points is changed regarding data connection.

**Requirement**

- A "WinCC FunctionTrendControl" with the name "Control1" is inserted in a process picture in Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.
- You have already configured a user archive in your project. Or you are using the demo project from which we have taken the user archive for the example.
'VBS363
Sub OnClick(ByVal Item)
Dim objFXControl
Dim objTrendWindow
Dim objTrend
Dim objXAxis
Dim objYAxis
Dim startID
Dim FXServerDataX(3)
Dim FXServerDataY(3)
' create reference to FXControl
Set objFXControl = ScreenItems("Control1")
' create reference to new window, x and y axis
Set objTrendWindow = objFXControl.GetTrendWindowCollection.AddItem("myWindow")
Set objXAxis = objFXControl.GetXAxisCollection.AddItem("myXAxis")
Set objYAxis = objFXControl.GetYAxisCollection.AddItem("myYAxis")
' assign x and y axis to the window
objXAxis.TrendWindow = objTrendWindow.Name
objYAxis.TrendWindow = objTrendWindow.Name
' add new trend
Set objTrend = objFXControl.GetTrendCollection.AddItem("myTrend1")
' configure trend data connection (UserArchive)
objTrend.Provider = 3
startID = CLng(4)
FXServerDataX(0) = "Setpoint"
FXServerDataX(1) = "ParabelX"
FXServerDataX(3) = startID
FXServerDataY(0) = "Setpoint"
FXServerDataY(1) = "ParabelY"
FXServerDataY(3) = startID
objTrend.MeasurePoints = 50
objTrend.SetTagName "Setpoint\ParabelX", "Setpoint\ParabelY", FXServerDataX, FXServerDataY
' assign trend properties
objTrend.Color = RGB(255,0,0)
objTrend.PointStyle = 1
objTrend.TrendWindow = objTrendWindow.Name
objTrend.XAxis = objXAxis.Name
objTrend.YAxis = objYAxis.Name
End Sub

Note
More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".

See also
Controls (Page 218)
ServerDataX (Page 569)
ServerDataY (Page 570)
WinCC Gauge Control

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Gauge Control"

Type Identifier in VBS

HMIGauge

Usage

In the following example, the object with the name "Control1" is moved 14 pixels to the right:

'VBS58
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +14

See also

WarningColor Property (Page 697)
Object Property (Page 512)
BackColor Property (Page 322)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Object types of the ScreenItem object (Page 143)
Width Property (Page 698)
Warning Property (Page 697)
Visible Property (Page 696)
ValueMin Property (Page 695)
ValueMax Property (Page 695)
ValueColumnAlignment Property (Page 688)
UnitText Property (Page 672)
UnitOffset Property (Page 672)
UnitFont Property (Page 671)
UnitColor Property (Page 671)
Type Property (Page 666)
Top Property (Page 643)
TicWidth Property (Page 601)
TicTextOffset Property (Page 600)
TicTextColor Property (Page 600)
TicOffset Property (Page 600)
TicFont Property (Page 599)
TicColor Property (Page 599)
ShowWarning Property (Page 579)
ShowPeak Property (Page 574)
ShowNormal Property (Page 573)
ShowDecimalPoint Property (Page 573)
ShowDanger Property (Page 573)
Rectangular Property (Page 549)
Parent Property (Page 529)
ObjectName Property (Page 512)
NormalColor Property (Page 510)
NeedleColor Property (Page 510)
LocaleID Property (Page 482)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FrameScale Property (Page 437)
FramePicture Property (Page 437)
FrameColor Property (Page 436)
Enabled Property (Page 408)
Delta Property (Page 402)
DangerColor Property (Page 397)
WinCC Media Control

Description

```
Objects

<table>
<thead>
<tr>
<th>ScreenItems</th>
<th>ScreenItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screens</td>
<td>Screen</td>
</tr>
</tbody>
</table>
```

Object Type of ScreenItem Object. Represents the "WinCC Media Control" graphic object as of WinCC V7.0.

Object Type of ScreenItem Object. Represents the "WinCC Media Control" graphic object as of WinCC V7.0.

Type Identifier in VBS

HMIMediaControl
Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```vbs
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

WinCC OnlineTableControl

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC OnlineTableControl" as of WinCC V7.0.

Type Identifier in VBS

HMIOnlineTableControl

Available list objects

<table>
<thead>
<tr>
<th>Row (Page 225)</th>
<th>ToolbarButton (Page 232)</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusbarElement (Page 229)</td>
<td>ValueColumn (Page 236)</td>
</tr>
<tr>
<td>TimeColumn (Page 231)</td>
<td></td>
</tr>
</tbody>
</table>

Methods Available in VBS

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic</th>
<th>AttachDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CalculateStatistic</td>
<td>CopyRows</td>
<td></td>
</tr>
</tbody>
</table>
Properties Available in VBS

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "timecolobj.TimeColumnName", the listing name "TimeColumn" is dropped: "timecolobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>AutoCompleteColumns</th>
<th>AutoSelectionColors</th>
<th>AutoSelectionRectColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCompleteRows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackColor</td>
<td>BorderColor</td>
<td>BorderWidth</td>
</tr>
</tbody>
</table>
### VBS for Creating Procedures and Actions

#### 1.14 VBS Reference

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td></td>
</tr>
<tr>
<td>CellSpaceRight</td>
<td></td>
</tr>
<tr>
<td>CellSpaceTop</td>
<td></td>
</tr>
<tr>
<td>CellSpaceBottom</td>
<td>Closeable</td>
</tr>
<tr>
<td>CellSpaceLeft</td>
<td>ColumnResize</td>
</tr>
<tr>
<td>EnableEdit</td>
<td>ExportFilename</td>
</tr>
<tr>
<td>ExportDirectoryChangeable</td>
<td>ExportFilenameChangeable</td>
</tr>
<tr>
<td>ExportDirectoryname</td>
<td>ExportFormatGuid</td>
</tr>
<tr>
<td>ExportFileExtension</td>
<td>ExportFormatName</td>
</tr>
<tr>
<td>Font</td>
<td></td>
</tr>
<tr>
<td>GridLineColor</td>
<td>GridLineWidth</td>
</tr>
<tr>
<td>HorizontalGridLines</td>
<td></td>
</tr>
<tr>
<td>IconSpace</td>
<td></td>
</tr>
<tr>
<td>LineColor</td>
<td>LineWidth</td>
</tr>
<tr>
<td>Moveable</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>PrintJobName</td>
<td></td>
</tr>
<tr>
<td>RowCellCount (Page 364)</td>
<td>RowScrollbar</td>
</tr>
<tr>
<td>RowCellText (Page 366)</td>
<td>RowTitleAlign</td>
</tr>
<tr>
<td>RowCount (Page 394)</td>
<td>RowTitles</td>
</tr>
<tr>
<td>RowNumber (Page 553)</td>
<td>RTPersistence</td>
</tr>
<tr>
<td>RTPersistencePasswordLevel</td>
<td></td>
</tr>
<tr>
<td>RTPersistenceType</td>
<td></td>
</tr>
</tbody>
</table>
### 1.14 VBS Reference

<table>
<thead>
<tr>
<th>SelectedCellColor</th>
<th>SelectedColoring</th>
<th>ShowSortIcon</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectedCellForeColor</td>
<td>SelectedRect</td>
<td>ShowSortIndex</td>
</tr>
<tr>
<td>SelectedRowColor</td>
<td>SelectedRectColor</td>
<td>ShowTitle</td>
</tr>
<tr>
<td>SelectedRowForeColor</td>
<td>SelectedRectWidth</td>
<td>Sizeable</td>
</tr>
<tr>
<td>SelectedTitleColor</td>
<td>SelectionType</td>
<td>SkinName</td>
</tr>
<tr>
<td>SelectedTitleForeColor</td>
<td>ShowSortButton</td>
<td>SortSequence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StatusbarBackColor</th>
<th>StatusbarElementName</th>
<th>StatusbarElementVisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusbarElementAdd</td>
<td>StatusbarElementRemove</td>
<td>StatusbarElementWidth</td>
</tr>
<tr>
<td>StatusbarElementAutoSize</td>
<td>StatusbarElementRename</td>
<td>StatusbarFontColor</td>
</tr>
<tr>
<td>StatusbarElementCount</td>
<td>StatusbarElementRepos</td>
<td>StatusbarShowTooltips</td>
</tr>
<tr>
<td>StatusbarElementIconId</td>
<td>StatusbarElementText</td>
<td>StatusbarText</td>
</tr>
<tr>
<td>StatusbarElementId</td>
<td>StatusbarElementTooltipText</td>
<td>StatusbarUseBackColor</td>
</tr>
<tr>
<td>StatusbarElementIndex</td>
<td>StatusbarElementUserDefined</td>
<td>StatusbarVisible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TableColor</th>
<th>TableForeColor</th>
<th>TimeBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableColor2</td>
<td>TableForeColor2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TimeColumnActualize</th>
<th>TimeColumnHideText</th>
<th>TimeColumnShowDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeColumnAdd</td>
<td>TimeColumnHiddenText</td>
<td>TimeColumnShowIcon</td>
</tr>
<tr>
<td>TimeColumnAlign</td>
<td>TimeColumnIndex</td>
<td>TimeColumnShowTitleIcon</td>
</tr>
<tr>
<td>TimeColumnBackColor</td>
<td>TimeColumnLength</td>
<td>TimeColumnSort</td>
</tr>
<tr>
<td>TimeColumnBeginTime</td>
<td>TimeColumnMeasurePoints</td>
<td>TimeColumnSortIndex</td>
</tr>
<tr>
<td>TimeColumnCaption</td>
<td>TimeColumnName</td>
<td>TimeColumnSortFormat</td>
</tr>
<tr>
<td>TimeColumnCount</td>
<td>TimeColumnRangeType</td>
<td>TimeColumnTimeRangeBase</td>
</tr>
<tr>
<td>TimeColumnDateFormat</td>
<td>TimeColumnRemove</td>
<td>TimeColumnTimeRangeFactor</td>
</tr>
<tr>
<td>TimeColumnEndTime</td>
<td>TimeColumnRename</td>
<td>TimeColumnUseValueColumnColors</td>
</tr>
<tr>
<td>TimeColumnForeColor</td>
<td>TimeColumnRepos</td>
<td>TimeColumnVisible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TimeStepBase</th>
<th>TitleDarkShadowColor</th>
<th>TitleLightShadowColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeStepFactor</td>
<td>TitleForeColor</td>
<td>TitleSort</td>
</tr>
<tr>
<td>TitleColor</td>
<td>TitleGridLineColor</td>
<td>TitleStyle</td>
</tr>
<tr>
<td>TitleCut</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ToolbarAlignment</th>
<th>ToolbarButtonHotKey</th>
<th>ToolbarButtonRepos</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarBackColor</td>
<td>ToolbarButtonId</td>
<td>ToolbarButtonTooltipText</td>
</tr>
<tr>
<td>ToolbarBackColor</td>
<td>ToolbarButtonIndex</td>
<td>ToolbarButtonUserDefined</td>
</tr>
</tbody>
</table>
An additional column is added in an existing WinCC OnlineTableControl that is linked with an archive tag. Different properties are configured for the control and the column in the script.

**Requirement**

- A "WinCC OnlineTableControl" with the name "Control1" has already been inserted in a process picture in Graphics Designer. The control consists of a time column and three value columns. The picture "B_025_V7_Arch_TableControl.PDL" from the demo project was used for this example.

- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.

- You have already configured archives and archive tags in your project. Or you are using the demo project from which we have taken the archive for the example.
'VBS362
Sub OnClick(ByVal Item)
Dim objControl
Dim objTimeColumn
Dim objValueColumn
Set objControl = ScreenItems("Control1")
' Control wide specification
objControl.ColumnResize = False
objControl.TimeBase = 1
objControl.TimeColumnTimeFormat = "HH:mm:ss tt"
objControl.TimeColumnLength = 20
' properties for Time column
Set objTimeColumn = objControl.GetTimeColumn("Time column 1")
objTimeColumn.DateFormat = "dd/MM/yy"
' properties for a new 4th value column with connection to archive tag "Trend_4"
Set objValueColumn = objControl.GetValueColumnCollection.AddItem("Trend 4")
objValueColumn.Caption = "Trend 4"
objValueColumn.Length = 10
objValueColumn.Align = 1
objValueColumn.Provider = 1
objValueColumn.TagName = "G_Archive\Trend_4"
objValueColumn.TimeColumn = "Time column 1"
End Sub

Note
More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".

See also
Controls (Page 218)

WinCC OnlineTrendControl

Description

Screens
Screen
ScreenItems
ScreenItem
Object Type of ScreenItem Object. Represents the graphic object "WinCC OnlineTrendControl" as of WinCC V7.0.

**Type Identifier in VBS**

HMIOnlineTrendControl

**Available list objects**

<table>
<thead>
<tr>
<th>StatusbarElement (Page 229)</th>
<th>ToolBarButton (Page 232)</th>
<th>TrendWindow (Page 234)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeAxis (Page 230)</td>
<td>Trend (Page 233)</td>
<td>ValueAxis (Page 235)</td>
</tr>
</tbody>
</table>

**Methods Available in VBS**

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic-Methode</th>
<th>AttachDB-Methode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CalculateStatistic</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeactivateDynamic</td>
<td>DetachDB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetStatusBarElement</td>
<td>GetToolBarButton (Page 752)</td>
<td>GetTrendWindow</td>
</tr>
<tr>
<td>GetStatusBarElementCollection</td>
<td>GetToolBarButtonCollection (Page 753)</td>
<td>GetTrendWindowCollection</td>
</tr>
<tr>
<td>GetTimeAxis</td>
<td>GetTrend</td>
<td>GetValueAxis</td>
</tr>
<tr>
<td>GetTimeAxisCollection</td>
<td>GetTrendCollection</td>
<td>GetValueAxisCollection</td>
</tr>
<tr>
<td>MoveAxis</td>
<td>MoveToLast</td>
<td>MoveToPrevious</td>
</tr>
<tr>
<td>MoveToFirst</td>
<td>MoveToNext</td>
<td></td>
</tr>
<tr>
<td>NextTrend</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OneToOneView</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreviousTrend</td>
<td>Print</td>
<td></td>
</tr>
</tbody>
</table>
### Properties Available in VBS

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "trendobj.Trendname", the listing name "Trend" is dropped: "trendobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>BackColor</th>
<th>BorderColor</th>
<th>BorderWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td>Closeable</td>
<td>ConnectTrendWindows</td>
</tr>
<tr>
<td>ExportDirectoryChangeable</td>
<td>ExportFilenameChangeable</td>
<td>ExportSelection</td>
</tr>
<tr>
<td>ExportDirectoryName</td>
<td>ExportFormatGuid</td>
<td>ExportShowDialog</td>
</tr>
<tr>
<td>ExportFileExtension</td>
<td>ExportFormatName</td>
<td>ExportXML</td>
</tr>
<tr>
<td>ExportFilename</td>
<td>ExportParameters</td>
<td></td>
</tr>
<tr>
<td>Font</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphDirection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LineColor</td>
<td>LineWidth</td>
<td>LoadDataImmediately</td>
</tr>
<tr>
<td>Moveable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PercentageAxis</td>
<td>PercentageAxisColor</td>
<td>PrintJobName</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>PercentageAxisAlign</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RTPersistence</th>
<th>RTPersistencePasswordLevel</th>
<th>RTPersistenceType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ShowRuler</th>
<th>ShowStatisticRuler</th>
<th>Sizeable</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowRulerInAxis</td>
<td>ShowTitle</td>
<td>SkinName</td>
</tr>
<tr>
<td>ShowScrollbars</td>
<td>ShowTrendIcon</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StatusBarBackColor</th>
<th>StatusBarElementName</th>
<th>StatusBarElementVisible</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusBarElementAdd</td>
<td>StatusBarElementRemove</td>
<td>StatusBarElementWidth</td>
</tr>
<tr>
<td>StatusBarElementAutoSize</td>
<td>StatusBarElementRename</td>
<td>StatusBarFontColor</td>
</tr>
<tr>
<td>StatusBarElementCount</td>
<td>StatusBarElementRepos</td>
<td>StatusBarShowToolTips</td>
</tr>
<tr>
<td>StatusBarElementIconId</td>
<td>StatusBarElementText</td>
<td>StatusBarText</td>
</tr>
<tr>
<td>StatusBarElementId</td>
<td>StatusBarElementTooltipText</td>
<td>StatusBarUseBackColor</td>
</tr>
<tr>
<td>StatusBarElementIndex</td>
<td>StatusBarElementUserDefined</td>
<td>StatusBarVisible</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TimeAxisActualize</th>
<th>TimeAxisIndex</th>
<th>TimeAxisRepos</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeAxisAdd</td>
<td>TimeAxisInTrendColor</td>
<td>TimeAxisShowDate</td>
</tr>
<tr>
<td>TimeAxisAlign</td>
<td>TimeAxisLabel</td>
<td>TimeAxisTimeFormat</td>
</tr>
<tr>
<td>TimeAxisBeginTime</td>
<td>TimeAxisMeasurePoints</td>
<td>TimeAxisTimeRangeBase</td>
</tr>
<tr>
<td>TimeAxisColor</td>
<td>TimeAxisName</td>
<td>TimeAxisTimeRangeFactor</td>
</tr>
<tr>
<td>TimeAxisCount</td>
<td>TimeAxisRangeType</td>
<td>TimeAxisTrendWindow</td>
</tr>
<tr>
<td>TimeAxisDateFormat</td>
<td>TimeAxisRemove</td>
<td>TimeAxisVisible</td>
</tr>
<tr>
<td>TimeAxisEndTime</td>
<td>TimeAxisRename</td>
<td>TimeBase</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ToolbarAlignment</th>
<th>ToolbarButtonHotKey</th>
<th>ToolbarButtonRepos</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarBackColor</td>
<td>ToolbarButtonId</td>
<td>ToolbarButtonTooltipText</td>
</tr>
<tr>
<td>ToolbarButtonActive</td>
<td>ToolbarButtonIndex</td>
<td>ToolbarButtonUserDefined</td>
</tr>
<tr>
<td>ToolbarButtonAdd</td>
<td>ToolbarButtonLocked</td>
<td>ToolbarButtonVisible</td>
</tr>
<tr>
<td>ToolbarButtonBeginGroup</td>
<td>ToolbarButtonName</td>
<td>ToolbarShowToolTips</td>
</tr>
<tr>
<td>ToolbarButtonClick</td>
<td>ToolbarButtonPasswordLevel</td>
<td>ToolbarUseBackColor</td>
</tr>
<tr>
<td>ToolbarButtonCount</td>
<td>ToolbarButtonRemove</td>
<td>ToolbarUseHotKeys</td>
</tr>
<tr>
<td>ToolbarButtonEnabled</td>
<td>ToolbarButtonRename</td>
<td>ToolbarVisible</td>
</tr>
</tbody>
</table>
### Example

Three trends are displayed in a WinCC OnlineTrendControl that are linked with archive tags. Different properties are configured for the trends in the script.
Requirement

- A "WinCC OnlineTrendControl" with the name "Control1" is inserted in a process picture in Graphics Designer.

- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.

- You have already configured archives and archive tags in your project. Or you are using the demo project from which we have taken the archives for the example.
'VBS361
Sub OnClick(ByVal Item)
Dim objTrendControl
Dim objTrendWindow
Dim objTimeAxis
Dim objValueAxis
Dim objTrend
' create reference to TrendControl
Set objTrendControl = ScreenItems("Control1")
' create reference to new window, time and value axis
Set objTrendWindow = objTrendControl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = objTrendControl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValueAxis = objTrendControl.GetValueAxisCollection.AddItem("myValueAxis")
' assign time and value axis to the window
objTimeAxis.TrendWindow = objTrendWindow.Name
objValueAxis.TrendWindow = objTrendWindow.Name
' assign properties to trendwindow
objTrendWindow.HorizontalGrid = False
' add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "G_Archive\Trend_1"
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name
objTrend.Color = RGB(255,0,0)
objTrend.PointStyle = 0
' add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend2")
objTrend.Provider = 1
objTrend.TagName = "G_Archive\Trend_2"
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name
objTrend.Color = RGB(0,255,0)
objTrend.LineWidth = 3
' add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend3")
objTrend.Provider = 1
objTrend.TagName = "G_Archive\Trend_3"
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name
objTrend.Color = RGB(0,0,255)
objTrend.LineType = 2
End Sub

Note
More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".
See also

Controls (Page 218)

WinCC Push Button Control

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Push Button Control"

Type Identifier in VBS

HMIButton

Usage

In the following example, the object with the name "Control1" is moved 17 pixels to the right:

```vbs
'VBS61
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +17
```

Note

The events KeyDown, KeyUp and KeyPress cannot be addressed by VBS. If it is required to make controls dynamic with the help of VBS, no parameter must be declared with ByRef.
Notes on Error Handling

Buttons and pushbuttons are mapped in the object model to an "HMIButton" type. Since the objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0

Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handling. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.
Examples of error handling

'VBS62
Dim objScreenItem
On Error Resume Next    'Activation of error handling
For Each objScreenItem In ScreenItems
    If objScreenItem.Type = "HMIButton" Then
        '=== Property "Text" available only for Standard-Button
        objScreenItem.Text = "Windows"
        If 0 <> Err.Number Then
            HMIRuntime.Trace objScreenItem.ObjectName & ": no Windows-Button" & vbCrLf
            Err.Clear    'Delete error message
        End If
        '=== Property "Caption" available only for PushButton
        objScreenItem.Caption = "Push"
        If 0 <> Err.Number Then
            HMIRuntime.Trace objScreenItem.ObjectName & ": no Control" & vbCrLf
            Err.Clear
        End If
    End If
Next
On Error Goto 0    'Deactivation of error handling

See also

- Properties (Page 301)
- FontName Property (Page 433)
- Activate Method (Page 712)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Controls (Page 218)
- Width Property (Page 698)
- Visible Property (Page 696)
- Type Property (Page 666)
- Transparent Property (Page 644)
- Top Property (Page 643)
- PictureUnselected Property (Page 539)
- PictureSelected Property (Page 538)
- Parent Property (Page 529)
- Outline Property (Page 527)
- ObjectName Property (Page 512)
Object Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
FrameWidth Property (Page 438)
FrameColorUp Property (Page 437)
FrameColorDown Property (Page 436)
ForeColor Property (Page 435)
FontUnderline Property (Page 434)
FontStrikeThru Property (Page 434)
FontSize Property (Page 433)
FontItalic Property (Page 432)
Font property (before WinCC V7) (Page 431)
FontBold Property (Page 432)
FocusRect Property (Page 430)
Enabled Property (Page 408)
Caption Property (Page 362)
BackColor Property (Page 322)
AutoSize Property (Page 319)

WinCC RulerControl

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC RulerControl" as of WinCC V7.0.

Type Identifier in VBS

HMIRulerControl
### Available list objects

<table>
<thead>
<tr>
<th>Row (Page 225)</th>
<th>StatisticResultColumn (Page 229)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RulerBlock (Page 226)</td>
<td>StatusbarElement (Page 229)</td>
</tr>
<tr>
<td>RulerColumn (Page 227)</td>
<td>ToolbarButton (Page 232)</td>
</tr>
<tr>
<td>StatisticAreaColumn (Page 228)</td>
<td></td>
</tr>
</tbody>
</table>

### Methods Available in VBS

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeactivateDynamic</td>
<td></td>
</tr>
<tr>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>

| GetRow (Page 731)                                                         | GetSelectedRow (Page 738)         | GetStatusbarElement            |
|                                                                          | GetSelectedRows (Page 739)        | GetStatusbarElementCollection  |
| GetRulerBlock                                                            | GetStatisticAreaColumn           | GetToolbarButton               |
| GetRulerBlockCollection                                                  | GetStatisticAreaColumnCollection | GetToolbarButtonCollection     |
| GetRulerColumn                                                          | GetStatisticResultColumn         |                                   |
| GetRulerColumnCollection                                                | GetStatisticResultColumnCollection |

<table>
<thead>
<tr>
<th>ShowHelp</th>
<th>ShowPropertyDialog</th>
</tr>
</thead>
</table>

### Properties Available in VBS

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "rulercolobj.ColumnName", the listing name "Column" is dropped: "rulercolobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>AutoCompleteColumns</th>
<th>AutoPosition</th>
<th>AutoSelectionRectColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCompleteRows</td>
<td>AutoSelectionColors</td>
<td>AutoShow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BackColor</th>
<th>BlockHideText</th>
<th>BlockShowDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockAlign</td>
<td>BlockHideTitleText</td>
<td>BlockShowIcon</td>
</tr>
<tr>
<td>BlockAutoPrecisions</td>
<td>BlockID</td>
<td>BlockShowTitleIcon</td>
</tr>
<tr>
<td>BlockCaption</td>
<td>BlockIndex</td>
<td>BlockTimeFormat</td>
</tr>
</tbody>
</table>
### VBS Reference

<table>
<thead>
<tr>
<th>BlockCount</th>
<th>BlockLength</th>
<th>BlockUseSourceFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockDateFormat</td>
<td>BlockName</td>
<td>BorderColor</td>
</tr>
<tr>
<td>BlockExponentialFormat</td>
<td>BlockPrecisions</td>
<td>BorderWidth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Caption</th>
<th>ColumnAdd</th>
<th>ColumnScrollbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellCut</td>
<td>ColumnCount</td>
<td>ColumnSort</td>
</tr>
<tr>
<td>CellSpaceBottom</td>
<td>ColumnIndex</td>
<td>ColumnSortIndex</td>
</tr>
<tr>
<td>CellSpaceLeft</td>
<td>ColumnName</td>
<td>ColumnTitleAlign</td>
</tr>
<tr>
<td>CellSpaceRight</td>
<td>ColumnRemove</td>
<td>ColumnTitles</td>
</tr>
<tr>
<td>CellSpaceTop</td>
<td>ColumnRepos</td>
<td>ColumnVisible</td>
</tr>
<tr>
<td>Closeable</td>
<td>ColumnResize</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ExportDirectoryChangeable</th>
<th>ExportFilenameChangeable</th>
<th>ExportSelection</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportDirectoryName</td>
<td>ExportFormatGuid</td>
<td>ExportShowDialog</td>
</tr>
<tr>
<td>ExportFileExtension</td>
<td>ExportFormatName</td>
<td>ExportXML</td>
</tr>
<tr>
<td>ExportFilename</td>
<td>ExportParameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Font</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GridLineColor</td>
<td>GridLineWidth</td>
</tr>
<tr>
<td>HorizontalGridLines</td>
<td></td>
</tr>
<tr>
<td>IconSpace</td>
<td></td>
</tr>
<tr>
<td>LineColor</td>
<td>LineWidth</td>
</tr>
<tr>
<td>Moveable</td>
<td></td>
</tr>
<tr>
<td>PrintJobName</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RowCellCount (Page 364)</th>
<th>RowScrollbar</th>
<th>RTPersistencePasswordLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowCellText (Page 366)</td>
<td>RowTitleAlign</td>
<td>RTPersistenceType</td>
</tr>
</tbody>
</table>
## 1.14 VBS Reference

<table>
<thead>
<tr>
<th><strong>RowCount (Page 394)</strong></th>
<th><strong>RowTitles</strong></th>
<th><strong>RulerType</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RowNumber (Page 553)</strong></td>
<td><strong>RTPersistence</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SelectedCellColor</strong></th>
<th><strong>SelectionRect</strong></th>
<th><strong>ShowSortIndex</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SelectedCellForeColor</strong></td>
<td><strong>SelectionRectColor</strong></td>
<td><strong>ShowTitle</strong></td>
</tr>
<tr>
<td><strong>SelectedRowColor</strong></td>
<td><strong>SelectionRectWidth</strong></td>
<td><strong>Sizeable</strong></td>
</tr>
<tr>
<td><strong>SelectedRowForeColor</strong></td>
<td><strong>SelectionType</strong></td>
<td><strong>SkinName</strong></td>
</tr>
<tr>
<td><strong>SelectedTitleColor</strong></td>
<td><strong>ShareSpaceWithSourceControl</strong></td>
<td><strong>SortSequence</strong></td>
</tr>
<tr>
<td><strong>SelectedTitleForeColor</strong></td>
<td><strong>ShowSortButton</strong></td>
<td><strong>SourceControl</strong></td>
</tr>
<tr>
<td><strong>SelectionColoring</strong></td>
<td><strong>ShowSortIcon</strong></td>
<td><strong>SourceControlType</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>StatusBarBackColor</strong></th>
<th><strong>StatusBarElementName</strong></th>
<th><strong>StatusBarElementVisible</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StatusBarElementAdd</strong></td>
<td><strong>StatusBarElementRemove</strong></td>
<td><strong>StatusBarElementWidth</strong></td>
</tr>
<tr>
<td><strong>StatusBarElementAutoSize</strong></td>
<td><strong>StatusBarElementRename</strong></td>
<td><strong>StatusBarFontColor</strong></td>
</tr>
<tr>
<td><strong>StatusBarElementCount</strong></td>
<td><strong>StatusBarElementRepos</strong></td>
<td><strong>StatusBarShowToolTips</strong></td>
</tr>
<tr>
<td><strong>StatusBarElementId</strong></td>
<td><strong>StatusBarElementText</strong></td>
<td><strong>StatusBarText</strong></td>
</tr>
<tr>
<td><strong>StatusBarElementIndex</strong></td>
<td><strong>StatusBarElementToolTipText</strong></td>
<td><strong>StatusBarUseBackColor</strong></td>
</tr>
<tr>
<td><strong>StatusBarElementUserDefined</strong></td>
<td><strong>StatusBarElementVisible</strong></td>
<td><strong>StatusBarVisible</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TableColor</strong></th>
<th><strong>TitleColor</strong></th>
<th><strong>TitleGridLineColor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TableColor2</strong></td>
<td><strong>TitleCut</strong></td>
<td><strong>TitleLightShadowColor</strong></td>
</tr>
<tr>
<td><strong>TableForeColor</strong></td>
<td><strong>TitleDarkShadowColor</strong></td>
<td><strong>TitleSort</strong></td>
</tr>
<tr>
<td><strong>TableForeColor2</strong></td>
<td><strong>TitleForeColor</strong></td>
<td><strong>TitleStyle</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ToolbarAlignment</strong></th>
<th><strong>ToolbarButtonHotKey</strong></th>
<th><strong>ToolbarButtonRepos</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToolbarBackColor</strong></td>
<td><strong>ToolbarButtonId</strong></td>
<td><strong>ToolbarButtonTooltipText</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonActive</strong></td>
<td><strong>ToolbarButtonIndex</strong></td>
<td><strong>ToolbarButtonUserDefined</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonAdd</strong></td>
<td><strong>ToolbarButtonLocked</strong></td>
<td><strong>ToolbarButtonVisible</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonBeginGroup</strong></td>
<td><strong>ToolbarButtonName</strong></td>
<td><strong>ToolbarShowTooltip</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonClick</strong></td>
<td><strong>ToolbarButtonPasswordLevel</strong></td>
<td><strong>ToolbarUseBackColor</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonCount</strong></td>
<td><strong>ToolbarButtonRemove</strong></td>
<td><strong>ToolbarUseHotKeys</strong></td>
</tr>
<tr>
<td><strong>ToolbarButtonEnabled</strong></td>
<td><strong>ToolbarButtonRename</strong></td>
<td><strong>ToolbarVisible</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>UseSelectedTitleColor</strong></th>
<th><strong>UseSourceForeColor</strong></th>
<th><strong>UseTableColor2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UseSourceBackColors</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>VerticalGridLines</strong></th>
<th></th>
</tr>
</thead>
</table>
Example

A WinCC Ruler Control is inserted in a picture with an existing WinCC OnlineTableControl. The RulerControl contains a statistics window that displays the "Minimum", "Maximum" and "Average" columns. The static values are then displayed for the selected rows of the OnlineTableControl.

Requirement

- A "WinCC OnlineTableControl" with the name "Control1" has already been inserted in a process picture in Graphics Designer. The control is linked with archive tags or process tags. The picture "B_025_V7_Arch_TableControl.PDL" from the demo project was used for this example.
- You have added an additional "WinCC RulerControl" with the name "Control2" in the picture.
- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.
- You have selected some rows in OnlineTableControl.

'VBS364
Sub OnClick(ByVal Item)
Dim objRulerControl
Dim objTableControl
Dim objstatColumn
Dim rows

Set objRulerControl = ScreenItems("Control2")
' Use Statistic-window
objRulerControl.RulerType = 2
objRulerControl.SourceControl = "Controll"
' In Statistic-window only columns "Name", "MinValue", MaxValue" and "Average" are shown
Set objstatColumn = objRulerControl.GetStatisticResultColumnCollection
objstatColumn.RemoveItem(4)
objstatColumn.RemoveItem(5)
objstatColumn.RemoveItem(6)
' Get the selected rows of tablecontrol and calculate statistic
Set objTrendControl = ScreenItems("Controll")
Set rows = objTableControl.SelectAll
objTableControl.CalculateStatistic
End Sub

Note

More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".

See also

Controls (Page 218)
WinCC Slider Control

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Slider Control"

Type Identifier in VBS

HMISlider

Usage

In the following example, the object with the name "Control1" is moved 19 pixels to the right:

'VBS63
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +19

Notes on Error Handling

Sliders and WinCC slider controls are mapped in the object model to an "HMISlider" type. Since the objects have different properties, the availability of the property (dynamic type compilation in Runtime) should be queried via an exception measure. The exception measure is activated for the corresponding procedure by the following instruction:

On Error Resume Next

The instruction causes the VBScript engine to initiate the follow-on command in the case of a Runtime error.

The error code can subsequently be checked using the Err object. In order to deactivate the handling of Runtime errors in scripts, use the following command:

On Error Goto 0
Handling errors always relates to the procedure layer. If a script in a procedure causes an error, VBScript checks whether an error handling measure is implemented in this layer. If not, control is transferred one layer up (to the calling procedure). If there is no error handling measure here either, the control is transferred yet another layer up. This continues until either the top module level is reached or the code for Runtime error handling is located. If the activation of the Runtime error handling fails, the control is transferred to the top level on the internal VBScript Runtime error handling. This opens an error dialog and stops the script.

The "On Error Resume Next" command can be installed on all layers (i.e. also in procedures). When the error handling measure is use, it can basically be determined whether the user is actually using the required implementation type.

In addition, it can be ensured that there is no termination of execution due to a faulty access to the object.

**Examples of error handling**

```vbs
Sub OnClick(Byval Item)
'VBS193
Dim ScreenItem
' activating error handling:
On Error Resume Next
For Each ScreenItem In ScreenItems
  If ScreenItem.Type = "HMISlider" Then
    '=== Property "BevelColorUp" only exists for a WinCC Slider Control
    ScreenItem.BevelColorUp = 1
    If (Err.Number <> 0) Then
      HMIRuntime.Trace(ScreenItem.ObjectName + ": no Windows-Slider" + vbCrLf)
      ' delete error message
      Err.Clear
    End If
  '=== Property "BorderStyle" only exists for a Windows-Slider
  ScreenItem.BorderStyle = 1
  If (Err.Number <> 0) Then
    HMIRuntime.Trace(ScreenItem.ObjectName + ": no WinCC Slider Control" + vbCrLf)
    Err.Clear
  End If
  Next
On Error GoTo 0  ' deactivating error handling
End Sub
```

**See also**

- PictureThumb Property (Page 538)
- BarFillColor Property (Page 329)
- Activate Method (Page 712)
- Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Controls (Page 218)
WithLabels Property (Page 701)
WithAxes Property (Page 701)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
TickStyle Property (Page 602)
ThumbBackColor Property (Page 599)
ShowThumb Property (Page 577)
ShowPosition Property (Page 574)
ShowBar Property (Page 572)
RangeMin Property (Page 548)
RangeMax Property (Page 548)
Position Property (Page 541)
PictureBack Property (Page 536)
Parent Property (Page 529)
OuterBevelWidth Property (Page 527)
OuterBevelStyle Property (Page 526)
ObjectName Property (Page 512)
Object Property (Page 512)
LocaleID Property (Page 482)
Left Property (Page 476)
Layer Object (Page 120)
LabelColor Property (Page 457)
InnerBevelWidth Property (Page 453)
InnerBevelStyle Property (Page 453)
InnerBevelOffset Property (Page 453)
Height Property (Page 443)
ForeColor Property (Page 435)
FontPosition Property (Page 433)
Font property (before WinCC V7) (Page 431)
FocusWidth Property (Page 430)
WinCC UserArchiveControl

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC UserArchiveControl" as of WinCC V7.0.

Type Identifier in VBS

HMISeedleSubControl

Available list objects

<table>
<thead>
<tr>
<th>Column (Page 221)</th>
<th>Column (Page 221)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row (Page 225)</td>
<td>StatusBarElement (Page 229)</td>
</tr>
<tr>
<td></td>
<td>ToolbarButton (Page 232)</td>
</tr>
</tbody>
</table>

Methods Available in VBS

<table>
<thead>
<tr>
<th>Activate</th>
<th>ActivateDynamic</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CopyRows</th>
<th>CutRows</th>
</tr>
</thead>
</table>
### Properties Available in VBS

If you access the properties with the listing object, you do not have to enter the name of the listing. For example, when using "colobj.ColumnName", the listing name "Column" is dropped: "colobj.Name".

Note that properties are available for WinCC controls that can have the effect of methods. These properties are characterized by the respective names, e.g. "Add", "Remove" or "Rename".

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName</td>
<td>AutoCompleteColumns AutoSelectionColors</td>
</tr>
<tr>
<td>ArchiveType</td>
<td>AutoCompleteRows AutoSelectionRectColor</td>
</tr>
<tr>
<td>BackColor</td>
<td>BorderColor BorderWidth</td>
</tr>
</tbody>
</table>
VBS for Creating Procedures and Actions

1.14 VBS Reference
<table>
<thead>
<tr>
<th>Moveable</th>
<th>PrintJobName</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowCellCount (Page 364)</td>
<td>RowScrollbar</td>
</tr>
<tr>
<td>RowCellText (Page 366)</td>
<td>RTPersistence</td>
</tr>
<tr>
<td>RowCount (Page 394)</td>
<td>RTPersistencePasswordLevel</td>
</tr>
<tr>
<td>RowNumber (Page 553)</td>
<td>RTPersistenceType</td>
</tr>
<tr>
<td>SelectArchiveName</td>
<td>SelectionColoring</td>
</tr>
<tr>
<td>SelectedCellColor</td>
<td>SelectionRect</td>
</tr>
<tr>
<td>SelectedCellForeColor</td>
<td>SelectionRectColor</td>
</tr>
<tr>
<td>SelectedRowColor</td>
<td>SelectionRectWidth</td>
</tr>
<tr>
<td>SelectedRowForeColor</td>
<td>SelectionType</td>
</tr>
<tr>
<td>SelectedTitleColor</td>
<td>ShowSortButton</td>
</tr>
<tr>
<td>SelectedTitleForeColor</td>
<td>SortSequence</td>
</tr>
<tr>
<td>StatusbarBackColor</td>
<td>StatusbarElementName</td>
</tr>
<tr>
<td>StatusbarElementAdd</td>
<td>StatusbarElementRemove</td>
</tr>
<tr>
<td>StatusbarElementAutoSize</td>
<td>StatusbarElementRename</td>
</tr>
<tr>
<td>StatusbarElementCount</td>
<td>StatusbarElementRepos</td>
</tr>
<tr>
<td>StatusbarElementIconId</td>
<td>StatusbarElementText</td>
</tr>
<tr>
<td>StatusbarElementId</td>
<td>StatusbarElementToolTipText</td>
</tr>
<tr>
<td>StatusbarElementIndex</td>
<td>StatusbarElementUserDefined</td>
</tr>
<tr>
<td>TableColor</td>
<td>TitleColor</td>
</tr>
<tr>
<td>TableColor2</td>
<td>TitleCut</td>
</tr>
<tr>
<td>TableForeColor</td>
<td>TitleDarkShadowColor</td>
</tr>
<tr>
<td>TableForeColor2</td>
<td>TitleForeColor</td>
</tr>
<tr>
<td>TimeBase</td>
<td>TitleStyle</td>
</tr>
<tr>
<td>ToolbarAlignment</td>
<td>ToolbarButtonHotKey</td>
</tr>
<tr>
<td>ToolbarBackColor</td>
<td>ToolbarButtonId</td>
</tr>
<tr>
<td>ToolbarButtonActive</td>
<td>ToolbarButtonIndex</td>
</tr>
<tr>
<td>ToolbarButtonAdd</td>
<td>ToolbarButtonLocked</td>
</tr>
<tr>
<td>ToolbarButtonBeginGroup</td>
<td>ToolbarButtonName</td>
</tr>
<tr>
<td>StatusbarElementVisible</td>
<td>StatusbarElementVisible</td>
</tr>
<tr>
<td>StatusbarElementName</td>
<td>StatusbarElementWidth</td>
</tr>
<tr>
<td>StatusbarElementRemove</td>
<td>StatusbarFontColor</td>
</tr>
<tr>
<td>StatusbarElementrepos</td>
<td>StatusbarShowToolTips</td>
</tr>
<tr>
<td>StatusbarElementText</td>
<td>StatusbarText</td>
</tr>
<tr>
<td>StatusbarElementToolTipText</td>
<td>StatusbarUseBackColor</td>
</tr>
<tr>
<td>StatusbarElementUserDefined</td>
<td>StatusbarVisible</td>
</tr>
<tr>
<td>TableGridLineColor</td>
<td>TitleGridColumnColor</td>
</tr>
<tr>
<td>TitleCut</td>
<td>TitleLightShadowColor</td>
</tr>
<tr>
<td>TitleDarkShadowColor</td>
<td>TitleSort</td>
</tr>
<tr>
<td>TitleForeColor</td>
<td>TitleStyle</td>
</tr>
</tbody>
</table>
### Example

A user archive is displayed in a WinCC UserArchiveControl.

The following actions are initiated via script:

- Selecting data
- Exporting data
- Printing a table

### Requirement

- A "WinCC UserArchiveControl" with the name "Control1" is inserted in a process picture in Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured the event "mouse click" with a VBS action and the following script for the button.
- You have already configured a user archive in your project. Or you are using the demo project from which you can use a user archive.
VBS365
Sub OnClick(ByVal Item)
Dim objUAControl
Dim objColumn
Dim coll
Dim field
' create reference to UserArchivControl
Set objUAControl = ScreenItems("Control1")
' Select user archive and general column properties
objUAControl.SelectArchiveName = True
objUAControl.ColumnResize = False
objUAControl.ColumnTitleAlign = 1
' properties for ID column
Set objColumn = objUAControl.GetColumn("ID")
objColumn.Length = 2
objColumn.Align = 0
' Select data
objUAControl.FilterSQL = "ID >=3"
' export the content as a CSV-file in the "ua" directory of the project folder
objUAControl.ServerExport
' print the control
objUAControl.PrintJobName = "UserArchiveControl - Table"
objUAControl.Print
End Sub

Note
More examples for use of properties and methods are available in the descriptions of the Get methods of the controls and under "Examples for VBScript/Examples in WinCC/Dynamizing controls".

See also
Controls (Page 218)
Controls before WinCC V7

WinCC Alarm Control (before WinCC V7)

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Alarm Control"

Type Identifier in VBS

HMIMessageView

Usage

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

'VBS54
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10

See also

ProjectPath Property (Page 545)
BackColor Property (Page 322)
Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Controls (Page 218)
WindowType Property (Page 701)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolbarButtons Property (Page 638)
Titleline Property (Page 626)
TitleCut property (before WinCC V7) (Page 625)
StatusbarPanes Property (Page 592)
ServerNames property (before WinCC V7) (Page 571)
SelectionType property (before WinCC V7) (Page 568)
SelectionRectWidth property (before WinCC V7) (Page 567)
SelectionRectColor property (before WinCC V7) (Page 567)
SelectionMode Property (Page 566)
PersistentRTPermission Property (Page 533)
PersistentRTCSPermission Property (Page 533)
Parent Property (Page 529)
ObjectName Property (Page 512)
Object Property (Page 512)
MsgFilterSQL property (before WinCC V7) (Page 508)
MsgCtrlFlags Property (Page 508)
LineTitle Property (Page 481)
LineHeight Property (Page 480)
LineFont Property (Page 480)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
HeaderSort Property (Page 443)
GridLineVert Property (Page 442)
GridLineHorz Property (Page 440)
Font property (before WinCC V7) (Page 431)
Enabled Property (Page 408)
ColWidth Property (Page 389)
ColTitle Property (Page 378)
ColMove Property (Page 374)
CellCut property (before WinCC V7) (Page 364)
Caption Property (Page 362)
WinCC Function Trend Control (before WinCC V7)

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Function Trend Control"

Type Identifier in VBS

HMIFunctionTrendView

Usage

In the following example, the object with the name "Control1" is moved 13 pixels to the right:

'VBS57
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +13
See also

Top Property (Page 643)
ScalingTypeY Property (Page 560)
Layer Object (Page 120)
DesiredCurveSourceUAArchive Property (Page 403)
Activate Method (Page 712)
Properties (Page 301)
ScreenIems Object (List) (Page 129)
ScreenItem Object (Page 125)
Controls (Page 218)
Width Property (Page 698)
Visible Property (Page 696)
UpperLimitValue Property (Page 674)
UpperLimit Property (Page 673)
UpperLimitColor Property (Page 674)
Type Property (Page 666)
ToolbarHotKeys Property (Page 640)
ToolbarButtons Property (Page 638)
ToolbarAlignment property (before WinCC V7) (Page 629)
Titleline Property (Page 626)
TimeZone Property (Page 624)
TimeAxisX Property (Page 609)
TagProviderClsid Property (Page 597)
SourceUAColumnY Property (Page 586)
SourceUAColumnX Property (Page 586)
SourceUAArchiveStartID Property (Page 585)
SourceUAArchive Property (Page 585)
SourceTimeRange Property (Page 585)
SourceTagProviderDataY Property (Page 584)
SourceTagProviderDataX Property (Page 584)
SourceTagNameY Property (Page 584)
SourceTagNameX Property (Page 583)
SourceNumberOfValues Property (Page 583)
SourceNumberOfUAValues Property (Page 582)
SourceEndTime Property (Page 582)
SourceBeginTime Property (Page 581)
ShowValuesExponentialY Property (Page 578)
ShowValuesExponentialX Property (Page 578)
ShowRulerImmediately Property (Page 575)
ScalingTypeX Property (Page 559)
RulerPrecisionY Property (Page 556)
RulerPrecisionX Property (Page 556)
Replacement Property (Page 550)
ReplacementColor Property (Page 551)
RelayCurves Property (Page 550)
ProviderType Property (Page 545)
PrecisionY Property (Page 542)
PrecisionX Property (Page 542)
PersistentRTPPermission Property (Page 533)
PersistentRT Property (Page 532)
PersistentRTCSPermission Property (Page 533)
PersistentRTCS Property (Page 532)
Parent Property (Page 529)
Online property (before WinCC V7) (Page 516)
ObjectName Property (Page 512)
Object Property (Page 512)
NumItems Property (Page 511)
Name Property (Page 509)
LowerLimitValue Property (Page 487)
LowerLimit Property (Page 486)
LowerLimitColor Property (Page 487)
LoadDataImmediately property (before WinCC V7) (Page 482)
Left Property (Page 476)
LabelY Property (Page 458)
LabelX Property (Page 457)
ItemVisible Property (Page 456)
InsertData Property (Page 454)
Index Property (Page 452)
Height Property (Page 443)
GridlinesY Property (Page 441)
GridlinesX Property (Page 441)
GridlinesValueY Property (Page 441)
GridlinesValueX Property (Page 440)
GraphDirection property (before WinCC V7) (Page 439)
FreezeProviderConnections Property (Page 438)
Font property (before WinCC V7) (Page 431)
FineGridY Property (Page 423)
FineGridX Property (Page 423)
FineGridValueY Property (Page 423)
FineGridValueX Property (Page 422)
EndY Property (Page 411)
EndX Property (Page 411)
Enabled Property (Page 408)
DesiredCurveVisible Property (Page 405)
DesiredCurveSourceUAColumnY Property (Page 404)
DesiredCurveSourceUAColumnX Property (Page 404)
DesiredCurveSourceUASubtitleArchiveStartID Property (Page 404)
DesiredCurveSourceNumberOfUAValues Property (Page 403)
DesiredCurveCurveForm Property (Page 403)
DesiredCurveColor Property (Page 402)
DeleteData Property (Page 402)
DataY Property (Page 400)
DataXY Property (Page 399)
DataX Property (Page 399)
DataIndex Property (Page 398)
CurveForm Property (Page 396)
CommonY Property (Page 391)
CommonX Property (Page 391)
Color Property (Page 374)
CoarseGridY Property (Page 372)
CoarseGridX Property (Page 372)
CoarseGridValueY Property (Page 373)
CoarseGridValueX Property (Page 373)
Closeable property (before WinCC V7) (Page 370)
Caption Property (Page 362)
WinCC Online Table Control (before WinCC V7)

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Online Table Control"

Type Identifier in VBS

HMITableView

Usage

In the following example, the object with the name "Control1" is moved 15 pixels to the right:

'VBS59
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +15
See also

- TimeOverlap Property (Page 620)
- ItemVisible Property (Page 456)
- PrintBackgroundColor Property (Page 543)
- Activate Method (Page 712)
- Properties (Page 301)
- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Controls (Page 218)
- Width Property (Page 698)
- Visible Property (Page 696)
- Variable Property (Page 695)
- ValueColumnAlignment Property (Page 688)
- UpperLimitValue Property (Page 674)
- UpperLimit Property (Page 673)
- UpperLimitColor Property (Page 674)
- Type Property (Page 666)
- Top Property (Page 643)
- ToolbarHotKeys Property (Page 640)
- Toolbar Property (Page 628)
- ToolbarButtons Property (Page 638)
- ToolbarAlignment property (before WinCC V7) (Page 629)
- Titleline Property (Page 626)
- TimeZone Property (Page 624)
- TimeRangeFactor Property (Page 622)
- TimeRange Property (Page 621)
- TimeRangeBase Property (Page 621)
- TimeOverlapColor Property (Page 620)
- TimeJump Property (Page 619)
- TimeJumpColor Property (Page 620)
- TimeFormat Property (Page 619)
- TimeColumnAlignment Property (Page 611)
- Statusbar Property (Page 588)
- PrintJob Property (Page 544)
- Precisions Property (Page 542)
PersistentRTPermission Property (Page 533)
PersistentRT Property (Page 532)
PersistentRTCSPermission Property (Page 533)
PersistentRTCS Property (Page 532)
Parent Property (Page 529)
Online property (before WinCC V7) (Page 516)
ObjectName Property (Page 512)
Object Property (Page 512)
NumItems Property (Page 511)
LowerLimitValue Property (Page 487)
LowerLimit Property (Page 486)
LowerLimitColor Property (Page 487)
LoadDataImmediately property (before WinCC V7) (Page 482)
Left Property (Page 476)
Layer Object (Page 120)
Index Property (Page 452)
Height Property (Page 443)
Font property (before WinCC V7) (Page 431)
EndTime Property (Page 410)
Enabled Property (Page 408)
Edit Property (Page 407)
Editable Property (Page 407)
CommonTime Property (Page 390)
Command Property (Page 390)
Color Property (Page 374)
Closeable property (before WinCC V7) (Page 370)
Caption Property (Page 362)
BeginTime Property (Page 343)
BackColor Property (Page 322)
Archive Property (Page 314)
AllowPersistence Property (Page 310)
Actualize Property (Page 305)
Activate property (before WinCC V7) (Page 302)
LocaleSpecificSettings Property (Page 483)
WinCC Online Trend Control (before WinCC V7)

Description

Object Type of ScreenItem Object. Represents the graphic object "WinCC Online Trend Control"

Type Identifier in VBS

HMITrendView

Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```vbs
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +16
```

See also

Properties (Page 301)
TimeAxis Property (Page 602)
LowerLimitColor Property (Page 487)
Caption Property (Page 362)
Activate Method (Page 712)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Controls (Page 218)
Width Property (Page 698)
Visible Property (Page 696)
UpperLimitValue Property (Page 674)
UpperLimit Property (Page 673)
UpperLimitColor Property (Page 674)
Type Property (Page 666)
Top Property (Page 643)
ToolBarHotKeys Property (Page 640)
ToolBar Property (Page 628)
ToolBarButtons Property (Page 638)
ToolBarAlignment property (before WinCC V7) (Page 629)
Titleline Property (Page 626)
TimeZone Property (Page 624)
TimeRangeFactor Property (Page 622)
TimeRange Property (Page 621)
TimeRangeBase Property (Page 621)
TimeOverlap Property (Page 620)
TimeOverlapColor Property (Page 620)
TimeJump Property (Page 619)
TimeJumpColor Property (Page 620)
TimeAxisFormat Property (Page 605)
TagName Property (Page 596)
Statusbar Property (Page 588)
ShowRulerImmediately Property (Page 575)
ServerData Property (Page 569)
RulerPrecisions Property (Page 556)
Replacement Property (Page 550)
ReplacementColor Property (Page 551)
RelayCurves Property (Page 550)
ProviderClsid Property (Page 545)
PrintJob Property (Page 544)
Precisions Property (Page 542)
PersistentRTPermission Property (Page 533)
PersistentRT Property (Page 532)
PersistentRTCSPermission Property (Page 533)
PersistentRTCS Property (Page 532)
Parent Property (Page 529)
Online property (before WinCC V7) (Page 516)
ObjectName Property (Page 512)
Object Property (Page 512)
NumItems Property (Page 511)
MeasurePoints Property (Page 494)
LowerLimitValue Property (Page 487)
LowerLimit Property (Page 486)
LoadDataImmediately property (before WinCC V7) (Page 482)
Left Property (Page 476)
Layer Object (Page 120)
Label Property (Page 457)
ItemVisible Property (Page 456)
Index Property (Page 452)
Height Property (Page 443)
GridLineValue Property (Page 442)
GridLines Property (Page 440)
GraphDirection property (before WinCC V7) (Page 439)
Font property (before WinCC V7) (Page 431)
FineGridValue Property (Page 422)
FineGrid Property (Page 422)
EndValue Property (Page 410)
EndTime Property (Page 410)
Enabled Property (Page 408)
CurveForm Property (Page 396)
CommonY Property (Page 391)
CommonX Property (Page 391)
Command Property (Page 390)
Color Property (Page 374)
CoarseGridValue Property (Page 372)
CoarseGrid Property (Page 371)
Closeable property (before WinCC V7) (Page 370)
BeginValue Property (Page 343)
BeginTime Property (Page 343)
BackColor Property (Page 322)
Autorange Property (Page 317)
AllowPersistence Property (Page 310)
Actualize Property (Page 305)
Activate property (before WinCC V7) (Page 302)
AdjustRuler Property (Page 307)
LineWidth property (before WinCC V7) (Page 481)
ScalingType Property (Page 558)
UseRangeSubstitutes Property (Page 676)
XAxisColor property (before WinCC V7) (Page 702)
HideTagNames Property (Page 445)
LocaleSpecificSettings Property (Page 483)
PrintBackgroundColor Property (Page 543)
ItemProviderClsid Property (Page 456)
OneY Property (Page 516)
AllowXAxisColor - Property (Page 311)
AnchorRuler Property (Page 312)
SavedTrend Property (Page 557)
SelectedTrend Property (Page 565)
ShowSpanNames Property (Page 577)
DefaultPrecision Property (Page 400)
DefaultRulerPrecision Property (Page 400)
LowerLimitTagName Property (Page 487)
UpperLimitTagName Property (Page 674)
UseOnlineTags Property (Page 676)

1.14.3.7 Customized Object

Description

Object Type of ScreenItem Object. Represents the graphic object "Customized Object".
Type Identifier in VBS

HMIScreenModule

Usage

You access customized properties in a customized object via the attribute name in VBS. Intellisense is only applicable to the customized object as a whole.

You will locate the attribute name under Properties of the properties placed outside (right-click Property) and can be modified there.

In the following example, the object with the name "CustomizedObject1" is moved 10 pixels to the right:

```vbs
'VBS65
Dim objCustomObject
Set objCustomObject = ScreenItems("CustomizedObject1")
objCustomObject.Left = objCustomObject.Left + 10
```

See also

Activate Method (Page 712)
Properties (Page 301)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Object types of the ScreenItem object (Page 143)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Parent Property (Page 529)
ObjectName Property (Page 512)
Left Property (Page 476)
Layer Object (Page 120)
Height Property (Page 443)
Enabled Property (Page 408)
1.14.3.8 Group

Description

Object Type of ScreenItem Object. Represents the graphic object "Group"

Type Identifier in VBS

HMIGroup

Usage

In the following example, the object with the name "Group1" is moved 10 pixels to the right:

'VBS66
Dim objGroup
Set objGroup = ScreenItems("Group1")
objGroup.Left = objGroup.Left + 10

See also

Properties (Page 301)
Activate Method (Page 712)
ScreenItems Object (List) (Page 129)
ScreenItem Object (Page 125)
Object types of the ScreenItem object (Page 143)
Width Property (Page 698)
Visible Property (Page 696)
Type Property (Page 666)
Top Property (Page 643)
ToolTipText Property (Page 641)
Parent Property (Page 529)
### 1.14.4 Properties

#### 1.14.4.1 Properties

#### Overview

The properties of the individual objects can be used to modify specific graphic objects and tags in Runtime, e.g. activating an operating element per mouse click or triggering a color change by modifying a tag value.

Properties on graphic objects can be addressed via the following syntax:

```
'VBS191
Dim obj
Set obj = ScreenItems("object1")
obj.property = Value
```

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

```
'VBS192
Dim obj
Set obj = ScreenItems("control1")
obj.Left = obj.Left + 10
```
AccessPath Property

Description
Displays the storage path (with hierarchy information) of a screen object (picture). The property corresponds to the full access code on the Screens Collections.

STRING (read only)

Example:
In the following example, the path of the picture "ScreenWindow1" is issued:

```
'VBS67
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.AccessPath
```

See also
ScreenItem Object (Page 125)
Screens Object (List) (Page 134)

Activate property (before WinCC V7)

Description
The data to be displayed is only requested from the archive server when this attribute is set. In order to reduce the picture opening times, this attribute should not be set and the value only dynamically changed when necessary.

Write/Read access

To differentiate between the "Activate" property form the "Activate" method, the property is accessed via "Object".
Example:

Dim ctrl
Set ctrl = ScreenItems("Control")
ctrl.Object.activate = true

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Activate property

Activate
The data to be displayed in the message window are only requested from the message server if you set this attribute. Instead of setting this attribute, it is advisable to change the value dynamically in order to reduce picture activation times.

To differentiate the "Activate" property from the "Activate" method, the property is addressed via "Object".

Example

Dim ctrlSet
ctrl = ScreenItems("Control")
ctrl.Object.activate = true

The attribute can be assigned dynamic properties by means of the name Activate. The data type is BOOLEAN.

ActiveProject Property

Description

Returns an object of type "Project".

See also

Path Property (Page 531)
Name Property (Page 509)
Ellipse segment (Page 147)
HMIRuntime Object (Page 118)
ActiveScreen Property

Description
Supplies a reference to the picture which contains the object with the current focus.

Usage
"ActiveScreen" is used in Runtime to address the properties of the picture which contains the currently focused object.

Example:
The following example assigns the name of the current picture to the tag "strScrName" and outputs it in a message:

```vbnet
'VBS68
Dim strScrName
strScrName = HMIRuntime.ActiveScreen.Objectname
MsgBox strScrName
```

See also
Screen Object (Page 131)
HMIRuntime Object (Page 118)

ActiveScreenItem Property

Description
Supplies a reference to the object currently in focus.

Usage
"ActiveScreenItem" is used in Runtime in order to address the properties of the object currently in focus.
Example:

The following example displays the name of the object in the "ScreenWindow1" picture which has the focus:

```vbs
'VBS69
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.ActiveScreenItem.ObjectName
```

See also

ScreenItem Object (Page 125)
HMIRuntime Object (Page 118)

Actualize Property

Description

The "Index" property references a column pair or a trend. "Actualize" defines whether a static or dynamic representation should be used for this column pair/trend.

- 0: Static display
- -1: Dynamic display

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

ActualPointLeft Property

Description

Defines or returns the x-coordinate of the current corner point in relation to the original picture (top left). Each corner point is identified by an index which is derived from the number ("PointCount") of corner point available.

A change of the value can affect the properties "Width" (object width) and "Left" (x-coordinate of the object position).
ActualPointTop Property

Description

Defines or returns the y-coordinate of the current corner point in relation to the original picture (top left). Each corner point is identified by an index which is derived from the number ("PointCount") of corner point available.

A change of the value can affect the properties "Height" (object height) and "Top" (y-coordinate of the position).

See also

Polyline (Page 158)
Polygon (Page 156)
ScreenItem Object (Page 125)

AdaptBorder Property

Description

TRUE, when the border should be dynamically adjusted to the size of the text. BOOLEAN write-read access.

For text list and I/O field: Read only access.

See also

Button (Page 201)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
I/O Field (Page 184)
ScreenItem Object (Page 125)
AdaptPicture Property

Description
Defines whether the picture displayed in a picture window should be adapted to the size of the picture window in Runtime or not. Read only access.
TRUE, when the picture adapts to the picture window size.
FALSE, when the picture does not adapt to the picture window size.

See also
Picture Window (Page 179)
ScreenItem Object (Page 125)

AdaptSize Property

Description
Defines whether the picture window should adapt to the size of the picture displayed in it during Runtime or not. Read only access.
TRUE, when the picture window adapts to the picture size.
FALSE, when the picture window does not adapt to the picture size.

See also
Picture Window (Page 179)
ScreenItem Object (Page 125)

AdjustRuler Property

Description
Specifies if the ruler window should be adjusted to the trend window upon each appearance.
TRUE, if you move the ruler window and make it appear and disappear again, it will be displayed in its original position and its original size.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)
Al - Ap

AlarmID property

Description
Returns the AlarmID of the Alarm object. The AlarmID is unique, and is assigned by the system.

AlarmID (readonly)

See also
Alarms object (list) (Page 110)

AlarmHigh Property

Description
Defines the top limit value at which an alarm should be triggered or returned. The type of the evaluation (in percent or absolute) is defined in the "TypeAlarmHigh" property. The "CheckAlarmHigh" property determines whether the monitoring for this limit value is activated.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

AlarmLogs Property

Description
Returns an object of type "AlarmLogs".

AlarmLogs (read-only)

See also
HMIRuntime Object (Page 118)
AlarmLow Property

Description
Defines the bottom limit value at which an alarm should be triggered or returned. The type of the evaluation (in percent or absolute) is defined in the "TypeAlarmLow" property. The "CheckAlarmLow" property determines whether the monitoring for this limit value is activated.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

Alignment Property

Description
Defines or returns the representation of the scale (left/right or top/bottom) according to the position of the bar graph object. The "Scaling" property must be set to TRUE for the scale to be displayed.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

AlignmentLeft Property

Description
Defines or returns the horizontal alignment of the text. Value range from 0 to 2.
0 = left
1 = centered
2 = right

See also
Group Display (Page 194)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
AlignmentTop Property

Description
Defines or returns the vertical alignment of the text. Value range from 0 to 2.
0 = top
1 = centered
2 = bottom

See also
Group Display (Page 194)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)

AllowPersistence Property

Description
TRUE, when settings regarding persistence are possible. BOOLEAN write-read access.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
AllowXAxisColor - Property

Description
TRUE if the defined color of the common X-axis is displayed in runtime. BOOLEAN write-read access.

AllServer property (before WinCC V7)

Description
Defines that the data to be displayed in the message window is required by all servers participating in a distributed system on which Alarm Logging is activated. Write/Read access.

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

AllServer property

All servers - AllServer
Selects all servers whose packages were loaded and on which "Alarm Logging Runtime" is activated in the startup list.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>All servers are activated.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Activates only the servers entered in &quot;Server selection&quot;.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name AllServer. The data type is BOOLEAN.

Analog Property

Description
TRUE, when the clock is to be displayed as an analog clock. BOOLEAN write-read access.

See also
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)
AnchorRuler Property

Description
TRUE if the ruler window is firmly linked to the curve window. BOOLEAN write-read access.

AngleAlpha Property

Description
Defines or returns depth angle α for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

AngleBeta Property

Description
Defines or returns depth angle β for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

AngleMax Property

Description
Defines or returns the angle on the scale at which the scale graduation ends. LONG write-read access.

The start and end of the scale graduation are described by the attributes "AngleMin" and "AngleMax" in angular degrees. AngleMin < AngleMax applies.

Angle 0 degrees is at the right side of the horizontal diameter of the graduated scale disk. Positive angle values are counted in a counterclockwise direction.
See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

AngleMin Property

Description
Defines or returns the angle on the scale at which the scale graduation begins. LONG write-read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

Application Property

Description
Returns the Graphics Designer application when the application property is used without an object identifier. If the application property is used with object identifier, it returns an application object which displays the application with which the defined object was created. Read only access.

See also

Application Window (Page 173)
ScreenItem Object (Page 125)

ApplyProjectSettings property

Apply project settings - ApplyProjectSettings
Activates the project settings derived from "Alarm Logging".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The &quot;Apply project settings&quot; check box is selected. The message blocks configured in &quot;Alarm Logging&quot; and their properties are activated in AlarmControl. The message blocks are displayed with these properties in the message window.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The &quot;Apply project settings&quot; check box is deactivated. You can add or remove message blocks, or edit their properties.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ApplyProjectSettings. The data type is BOOLEAN.
Archive Property

Description
The "Index" property references a pair of columns. "Archive" defines process archive values linked to the column pair. The name of the process value archive is specified in the following form: Server name::Archive name

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

ArchiveName property

Name - ArchiveName
Specifies the user archive or view to be displayed. Open the “Package Browser” dialog for configuring an archive or a view by clicking the button.

The attribute can be assigned dynamic properties by means of the name ArchiveName. The data type is STRING.

ArchiveType property

Type - ArchiveType
Specifies whether the selected user archive is an archive or a view. The field cannot be edited.

The attribute can be assigned dynamic properties by means of the name ArchiveType. The data type is LONG.

AspectRatio property

AspectRatio
Specifies if the aspect ratio is kept in movies.

The attribute can be assigned dynamic properties by means of the name AspectRatio. The data type is BOOLEAN.
Assignments Property

Description
A list which contains the assignments between the output values and the actual output texts to be output. The assignments depend on the set list type. The list type is defined with the ListType property. Read only access.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

AssumeOnExit Property

Description
TRUE, if the entered text is assumed upon exiting the entry field (e.g., with the key or mouse click). BOOLEAN write-read access.

See also
I/O Field (Page 184)
Text list (Page 196)
ScreenItem Object (Page 125)

AssumeOnFull Property

Description
TRUE, when the content of the input field is full (specified number of characters have been entered) and should be exited automatically and the input accepted. BOOLEAN write-read access.

See also
I/O Field (Page 184)
ScreenItem Object (Page 125)
AutoCompleteColumns property

Show empty columns - AutoCompleteColumns

Adds empty columns if the Control width is greater than the width of columns configured.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of empty columns.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of empty columns.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `AutoCompleteColumns`. The data type is BOOLEAN.

AutoCompleteRows property

Show empty rows - AutoCompleteRows

Enables the insertion of empty rows if the Control length is greater than the number of rows configured.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of empty rows.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of empty rows.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `AutoCompleteRows`. The data type is BOOLEAN.

AutoPosition property

Automatic positioning - AutoPosition

Defines whether to position the RulerControl exactly below the source control.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The RulerControl is positioned exactly below the source control.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The RulerControl is displayed in accordance with your configuration of the control position.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `AutoPosition`. The data type is BOOLEAN.
Autorange Property

Description
TRUE, when the value range of the Y-axis is determined automatically or defined by using the "BeginValue" and "EndValue" attributes. BOOLEAN write-read access.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

AutorangeX Property

Description
TRUE, when the value range of the X-axis is determined automatically. FALSE, when it is determined by means of the "BeginX" and "EndX" attributes. BOOLEAN write-read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

AutorangeY Property

Description
TRUE, when the value range of the Y-axis is determined automatically. FALSE, when it is determined by means of the "BeginY" and "EndY" attributes. BOOLEAN write-read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

AutoScroll property (before WinCC V7)

Description
Defines the behavior of the message window when a new message is received. BOOLEAN write-read access.
TRUE : A newly received message is appended to the list displayed in the message window and is automatically selected. The visible range of the message window is moved, if necessary.
FALSE: A newly received message is not selected. The visible range of the message window is not changed.

The targeted selection of messages is only possible when “AutoScroll” is not active.

The “AutoScroll” property is deactivated when the attribute “MsgCtrlFlag” = “-1” is set. This means that the most recent message is displayed at the top of the list in the message window.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

AutoScroll Property

Auto scrolling - AutoScroll

Defines the behavior of the message window after a new message events.

You can only select message lines if “Auto scrolling” is disabled.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>If “AutoScroll” is activated, a new activated message is appended to the list displayed in the message window and selected automatically. The visible area of the message window is shifted as required.</td>
</tr>
<tr>
<td>FALSE</td>
<td>New message events are not selected if “AutoScroll” is disabled. The visible area of the message window is not changed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name AutoScroll. The data type is BOOLEAN.

AutoSelectionColors property

Automatic selection coloring - AutoSelectionColor

Enables the display of default system colors as selection color for cells and rows.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The system colors are in use.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The custom colors are used.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name AutoSelectionColors. The data type is BOOLEAN.
AutoSelectionRectColor property

Automatic color assignment - AutoSelectionRectColor

Defines a system color for the selection border.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The system color is in use.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The custom color is used.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `AutoSelectionRectColors`. The data type is BOOLEAN.

AutoShow property

Show/hide automatically - AutoShow

Enables/disables automatic activation of the RulerControl on the display if you selected the button functions for the ruler, statistics range and for statistics in the source control.

The RulerControl is hidden again if you are no longer using the ruler, statistics range and statistics functions.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The RulerControl is displayed automatically.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The RulerControl is not displayed automatically.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `AutoShow`. The data type is BOOLEAN.

AutoSize Property

Description

Defines or returns the size adaptation of the object. The following values can be set:

- 0: No size adaptation.
- 1: The picture ("PictureSelected", "PictureUnselected" properties) is adapted to the button.
- 2: The button is adapted to the picture ("PictureSelected", "PictureUnselected" properties).

See also

- WinCC Push Button Control (Page 269)
- ScreenItem Object (Page 125)
Autostart property

Autostart

Specifies if movies are started automatically.
The attribute can be assigned dynamic properties by means of the name Autostart. The data type is BOOLEAN.

Average Property

Average

TRUE, if the mean value is calculated based on the last 10 values. A value change is conditional for calculation of a new mean value. The mean value is reset when you change a picture. If only one value is available when you change the picture, the following mean value is calculated: (5+0+0+0+0+0+0+0+0+0)/10=0,5.

BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

Axe Property

Description

Defines or returns the position of the 3D bar in the coordinate system. Value range from 0 to 2.

0: The 3D-bar is displayed on the X-axis.
1: The 3D-bar is displayed on the Y-axis.
2: The 3D-bar is displayed on the Z-axis.

See also

3D Bar (Page 169)
ScreenItem Object (Page 125)
**AxisSection Property**

**Description**
Defines or returns the distance between two long axis sections. The information on the distance is given in scale units and is dependent on the minimum and maximum values configured.

**See also**
Bar (Page 174)
ScreenItem Object (Page 125)

**1.14.4.3 B**

**Ba**

**BackBorderWidth Property**

**Description**
Defines or returns the width of the 3D border in pixels. The value for the width is dependent on the size of the object.

**See also**
ScreenItem Object (Page 125)
Button (Page 201)
Round Button (Page 209)
Slider (Page 212)
Group Display (Page 194)

**BackColor property**

**Background - BackColor**
Specifies the background color of the control. The button opens the "Color selection" dialog. The attribute can be assigned dynamic properties by means of the name **BackColor**. The data type is LONG.
BackColor property

**Background Color (BackColor)**

Specifies the icon background color in the "Color selection" dialog. The background color is displayed in "opaque" style.

The attribute can be assigned dynamic properties by means of the name `BackColor`. The data type is LONG.

BackColor Property

**Function**

Defines or returns the background color for the object.

For objects with a fill pattern, the background color is not displayed if "transparent" is defined as the fill style.

**Special features of the WinCC slider control**

The background color only takes effect when the object is at least partially filled.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Enter the appropriate decimal value for each of the three RGB values.

Example:

```
RGB(200, 150, 100)
```

**Example:**

The following example defines the background of the "ScreenWindow1" picture to red:

```
'VBS70
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.BackColor = RGB(255, 0, 0)
```

**See also**

FillStyle Property (Page 419)

FillColor Property (Page 418)

ScreenItem Object (Page 125)
BackColor2 Property

Description
Defines or returns the bar color for the display of the current value. LONG write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

BackColor3 Property

Description
Defines or returns the color of the bar background. LONG write-read access.

See also
ScreenItem Object (Page 125)
Bar (Page 174)

BackColorBottom Property

Description
Defines or returns the color for the bottom/right part of the slider. LONG write-read access.

See also
Slider (Page 212)
ScreenItem Object (Page 125)

BackColorTop Property

Description
Defines or returns the color for the top/left part of the slider. LONG write-read access.

See also
Slider (Page 212)
ScreenItem Object (Page 125)
BackFlashColorOff Property

Description
Defines or returns the color of the object background for the flash status "Off". LONG write-read access.

See also
ScreenItem Object (Page 125)

BackFlashColorOn Property

Description
Defines or returns the color of the object background for the flash status "On". LONG write-read access.

See also
ScreenItem Object (Page 125)

Background Property

Description
TRUE, when the background of the 3D-bar graph object should be visible. BOOLEAN write-read access.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

BackgroundPicture Property

Description
Returns the picture name of the background picture for the graduated scale disk. Read only access
See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

BackPictureAlignment property

Description
Defines or returns the mode of representation of the background image in the process picture.
LONG write-read access.

BackPictureName property

Description
Defines the path and file name of the background image in the process picture or returns it.
LONG write-read access.

BackStyle Property

Description

WinCC Digital/Analog Clock
Defines the type of background of the analog clock:
- 0: The rectangular background of the clock is filled by the specified background color.
- 1: The round numbered face of the clock is filled by the specified background color. This enables a round analog clock to be displayed.
- 2: Numbered face and rectangular background are transparent.

WinCC Gauge Control
Defines the type of background of the gauge:
- 0: The rectangular or square background of the gauge has a border color is filled with the specified color. The circular graduated scale disk is filled by the specified background color.
- 1: The rectangular or square background of the gauge is transparent. The circular graduated scale disk is filled by the specified background color. This enables a circular gauge to be displayed.
- 2: The rectangular or square background and graduated scale disk are transparent.
WinCC Slider Control
Defines whether the object background should be transparent.
- 0: The object background is not transparent
- 1: The object background is transparent

HMI Symbol Library
Defines the icon background transparency. Write/Read access.
- 0: The background is transparent and, thus, invisible.
- 1: The background is visible, the color of the background is defined by the "Background Color" attribute.

See also
- HMI Symbol Library (Page 239)
- WinCC Slider Control (Page 277)
- WinCC Gauge Control (Page 254)
- WinCC Digital/Analog Clock (Page 246)
- ScreenItem Object (Page 125)

Bar

BarAdd property

New - BarAdd
Creates a new diagram.
The attribute can be assigned dynamic properties using the name BarAdd. The data type is STRING.

BarBackColor Property

Description
Defines the background color in the area of the slider. The area stretches form "RangeMin" to "RangeMax".

See also
- WinCC Slider Control (Page 277)
- ScreenItem Object (Page 125)
BarBarGradientColor property

Fill pattern color - BarBarGradientColor

The "Fill Pattern Color" attribute defines the color of the pattern that is displayed on a filled area.

The attribute can be assigned dynamic properties using the name BarBarGradientColor. The data type is LONG.

BarBarGradientPattern property

Fill pattern - BarBarGradientPattern

The "Fill Pattern" attribute specifies the pattern for the display of bars.

So that the fill pattern becomes visible, the color of the fill pattern must differ from the background color.

49 fill patterns are available for selection:
- Fill pattern 1 "Solid" fills the object with the set background color.
- Fill pattern 2 "Transparent" means neither a background nor a fill pattern is displayed.

The attribute can be assigned dynamic properties using the name BarBarGradientPattern. The data type is LONG.

BarBarWindow property

Diagram window - BarBarWindow

Specifies the diagram window in which the selected diagram will be displayed. You specify the available diagram windows on the "Diagram window" tab.

The attribute can be assigned dynamic properties using the name BarWindow. The data type is STRING.

BarChartType property

Diagram type - BarChartType

Specifies how the diagram will be displayed.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bar diagram</td>
<td>A diagram with bars will be displayed.</td>
</tr>
<tr>
<td>1</td>
<td>Bars with value</td>
<td>In addition to the bars, the values will be displayed in the diagram. The display of values conforms to the configuration of the &quot;Alignment&quot; property and the write direction of the bars.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties using the name `BarChartType`. The data type is LONG.

**BarColor property**

**Border color - BarColor**

Indicates the border color of the bar. Use the button to open the "Color selection" dialog to select the color.

The attribute can be assigned dynamic properties using the name `BarColor`. The data type is LONG.

**BarCount property**

**BarCount**

Specifies the number of configured diagrams.

The attribute can be assigned dynamic properties using the name `BarCount`. The data type is LONG.

**BarDepth Property**

**Description**

Defines or returns the depth of the bar in pixels.

**See also**

3D Bar (Page 169)

ScreenItem Object (Page 125)

**BarFillColor property**

**Bar color - BarFillColor**

Indicates the fill color of the bars. For the "Bars with value" diagram type, the text background color is specified.

Use the button to open the "Color selection" dialog to select the color.

The attribute can be assigned dynamic properties using the name `BarFillColor`. The data type is LONG.
BarFillColor Property

Description
Defines the fill color in the area of the slider. The area stretches from "RangeMin" to the position of the slider.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

BarHeight Property

Description
Defines or returns the height of the bar in pixels.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

BarIndex property

BarIndex
References a configured diagram. Using this attribute, you can assign the values of other attributes to a specific diagram. The index must always be set before you change the properties of a diagram in Runtime.

Valid values for "BarIndex" are between 0 and "BarCount" minus 1. The "BarCount" attribute specifies the number of configured diagrams.

The "BarIndex" attribute can be assigned dynamic properties using the attribute BarRepos. The data type is LONG.

BarLabel property

Label - BarLabel
Specifies the label of the selected diagram. The label is displayed in Runtime when the "UseBarNameAsLabel" attribute has the value "FALSE".

The attribute can be assigned dynamic properties using the name BarLabel. The data type is STRING.
BarLineStyle property

Border style - BarLineStyle

Specifies which border type is used to represent the border of the bar.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Solid</td>
<td>The border is displayed as a solid border.</td>
</tr>
<tr>
<td>1</td>
<td>Dashed</td>
<td>The border is displayed as a dashed border.</td>
</tr>
<tr>
<td>2</td>
<td>Dotted</td>
<td>The border is displayed as a dotted line.</td>
</tr>
<tr>
<td>3</td>
<td>Dash dot</td>
<td>The border is displayed as a dash-dot line.</td>
</tr>
<tr>
<td>4</td>
<td>Dash-dot-dot</td>
<td>The border is displayed as a dash-dot-dot line.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarLineStyle`. The data type is LONG.

BarLineWidth property

Border width - BarLineWidth

Specifies the border width of the bar.

The attribute can be assigned dynamic properties using the name `BarLineWidth`. The data type is LONG.

BarLowerLimit property

BarLowerLimit

Specifies the low limit of a tag. If the tag value is less than "BarLowerLimit", the values are designated with the color set in "BarLowerLimitColor". This setting is enabled when the "BarLowerLimitColoring" attribute has the value "TRUE".

The attribute can be assigned dynamic properties using the name `BarLowerLimit`. The data type is DOUBLE.

BarLowerLimitColor property

BarLowerLimitColor

Specifies the color for designating tag values below the value at "BarLowerLimit". The setting is enabled when the "BarLowerLimitColoring" attribute has the value "TRUE".

The attribute can be assigned dynamic properties using the name `BarLowerLimitColor`. The data type is LONG.
BarLowerLimitColoring property

BarLowerLimitColoring

Specifies whether the "BarLowerLimitColor" attribute is used for designating tag values that are lower than the value at "BarLowerLimit".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The &quot;BarLowerLimitColor&quot; attribute is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The &quot;BarLowerLimitColor&quot; attribute is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarLowerLimitColoring`. The data type is BOOLEAN.

BarName property

Object name - BarName

Specifies the name of the selected diagram.

The "BarName" attribute can be assigned dynamic properties using the attribute `BarRename`.

BarProvider property

Data source - BarProvider

Specifies the data source of the selected diagram.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>No data source configured to create a connection in Runtime by means of a script.</td>
</tr>
<tr>
<td>1</td>
<td>Archive tags</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarProvider`.

BarProviderCLSID property

BarProviderCLSID

Displays the data source of the selected diagram.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No data source configured to create a connection in Runtime by means of a script.</td>
</tr>
<tr>
<td>(416A09D2-8B5A-11D2-8B81-006097A45D48)</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties using the name **BarProviderCLSID**. The data type is **STRING**.

**BarRemove property**

**Remove - BarRemove**

Removes the selected diagram from the list.

The attribute can be assigned dynamic properties using the name **BarRemove**. The data type is **STRING**.

**BarRename property**

**BarRename**

Renames the diagram that is referenced using the "BarIndex" attribute.

The attribute can be assigned dynamic properties using the name **BarRename**. You can use "BarRename" to also assign dynamic properties to the "BarName" attribute. The data type is **STRING**.

**BarRepos property**

**Up/down - BarRepos**

Changes the order of the selected diagrams in the diagram window. "Up" and "Down" move the selected diagrams up and down in the list. As a result, the diagram is displayed further in the foreground or background in runtime.

The attribute can be assigned dynamic properties using the name **BarRepos**. The data type is **LONG**.

**BarSelectTagName property**

**BarSelectTagName**

Opens a dialog for selecting the tag name for the data source of the diagram. Programmers can set this attribute to allow users to select a tag name using a button, for example.

The attribute can be assigned dynamic properties using the name **BarSelectTagName**. The data type is **BOOLEAN**.
BarTagName property

Tag name - BarTagName
Displays the tag name of the connected tag. With the button you open a dialog for selecting an archive tag.
The attribute can be assigned dynamic properties using the name BarTagName. The data type is STRING.

BarTimeAxis property

Time axis - BarTimeAxis
Specifies which time axis will be used for the selected diagram. You specify the available time axes on the "Time axes" tab.
The attribute can be assigned dynamic properties using the name BarTimeAxis. The data type is STRING.

BarUncertainColor property

BarUncertainColor
Values have an uncertain status if the initial value is unknown when Runtime starts or if a substitute value is used. You can use the "BarUncertainColor" attribute to specify the color used for identifying these values. The "BarUncertainColoring" attribute determines whether or not this setting is evaluated.
The attribute can be assigned dynamic properties using the name BarUncertainColor. The data type is LONG.

BarUncertainColoring property

BarUncertainColoring
Values have an uncertain status if the initial value is unknown when Runtime starts or if a substitute value is used. You can use the "BarUncertainColoring" attribute to enable identification of such values based on the color set in "BarUncertainColor".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The setting for the &quot;BarUncertainColor&quot; attribute is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The setting for the &quot;BarUncertainColor&quot; attribute is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name BarUncertainColoring. The data type is BOOLEAN.
BarUpperLimitColoring property

**BarUpperLimitColoring**

Specifies whether the "BarUpperLimitColor" attribute is used for designating tag values that are higher than the value at "BarUpperLimit".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The setting for the &quot;BarUpperLimitColor&quot; attribute is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The setting for the &quot;BarUpperLimitColor&quot; attribute is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name **BarUpperLimitColoring**. The data type is BOOLEAN.

BarUpperLimitColor property

**BarUpperLimitColor**

Specifies the color for designating tag values above the value at "BarUpperLimit". The setting is enabled when the "BarUpperLimitColoring" attribute has the value "TRUE" hat.

The attribute can be assigned dynamic properties using the name **BarUpperLimitColor**. The data type is LONG.

BarUpperLimit property

**BarUpperLimit**

Specifies the high limit of a tag. If the tag value is higher than "BarUpperLimit", the values are designated with the color set in "BarUpperLimitColor". This setting is enabled when the "BarUpperLimitColoring" attribute has the value "TRUE".

The attribute can be assigned dynamic properties using the name **BarUpperLimit**. The data type is DOUBLE.

BarValueAlignment property

**Alignment - BarValueAlignment**

Specifies the alignment of the displayed values for the "Bars with value" diagram type.

The following settings are available depending on the write direction of the bars:

- The write direction of bars is "From the right" or "From the left"
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bottom</td>
<td>The values are displayed at the bottom of the diagram window.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The values are displayed at the center of the diagram window.</td>
</tr>
<tr>
<td>2</td>
<td>Top</td>
<td>The values are displayed at the top of the diagram window.</td>
</tr>
</tbody>
</table>

- The write direction of bars is "From the top" or "From the bottom"

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Left</td>
<td>The values are displayed at the left of the diagram window.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The values are displayed at the center of the diagram window.</td>
</tr>
<tr>
<td>2</td>
<td>Right</td>
<td>The values are displayed at the right of the diagram window.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarValueAlignment`. The data type is LONG.

**BarValueAxis property**

**Value axis - BarValueAxis**

Specifies which value axis will be used for the selected diagram. You specify the available value axes on the "Value axes" tab.

The attribute can be assigned dynamic properties using the name `BarValueAxis`. The data type is STRING.

**BarValueUnit property**

**Unit - BarValueUnit**

Specifies a unit of the values that will be appended to the value to be displayed for the "Bars with value" diagram type. e.g. "%" or "°C"

The attribute can be assigned dynamic properties using the name `BarValueUnit`. The data type is STRING.

**BarVisible property**

**Diagram - BarVisible**

The list shows the diagrams you have created.

Select the diagrams in the list that you want to display in the diagram windows.

Click on a diagram in the list to adapt the properties and to assign axes and diagram windows to the diagram.

The attribute can be assigned dynamic properties using the name `BarVisible`. The data type is BOOLEAN.
BarWindowAdd property

New - BarWindowAdd

Creates a new diagram window.

The attribute can be assigned dynamic properties using the name BarWindowAdd. The data type is STRING.

BarWindowCoarseGrid property

Main scaling - BarWindowCoarseGrid

Enables the display of grid lines for the main scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of grid lines for the main scale.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of grid lines for the main scale.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name BarWindowCoarseGrid. The data type is BOOLEAN.

BarWindowCoarseGridColor property

Color main scale - BarWindowCoarseGridColor

Specifies the color of the grid lines for the main scale. Use the button to open the "Color selection" dialog to select the color.

The attribute can be assigned dynamic properties using the name BarWindowCoarseGridColor. The data type is LONG.

BarWindowCount property

BarWindowCount

Specifies the number of configured diagrams.

The attribute can be assigned dynamic properties using the name BarWindowCount. The data type is LONG.
BarWindowFineGrid property

Secondary scaling - BarWindowFineGrid
Specifies whether the grid lines are displayed for the secondary scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of grid lines for the secondary scale.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of grid lines for the secondary scale.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarWindowFineGrid`. The data type is `BOOLEAN`.

BarWindowFineGridColor property

Color secondary scale - BarWindowFineGridColor
Specifies the color of the grid lines for the secondary scale. Use the button to open the "Color selection" dialog to select the color.

The attribute can be assigned dynamic properties using the name `BarWindowFineGridColor`. The data type is `LONG`.

BarWindowForegroundBarGrid property

For foreground diagram only - BarWindowForegroundBarGrid
Specifies whether the grid lines for the foreground diagram only will be displayed in the selected diagram window.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The grid lines for the foreground diagram are displayed in the diagram window.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The grid lines for all diagrams are displayed in the diagram window.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarWindowForegroundBarGrid`. The data type is `BOOLEAN`.

BarWindowGapWidth property

Gap width - BarWindowGapWidth
Specifies the gap between two bars of a bar diagram in % of the bar width.

The attribute can be assigned dynamic properties using the name `BarWindowGapWidth`. The data type is `LONG`.
BarWindowGridInBarColor property

In diagram color - BarWindowGridInBarColor

Specifies whether the grid lines for the main scaling will be displayed in the color of the bars.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The grid lines are displayed in the color of the bars.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The grid lines are displayed with the color set in the &quot;Color&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarWindowGridInBarColor`. The data type is BOOLEAN.

BarWindowHorizontalGrid property

For value axes - BarWindowHorizontalGrid

Enables the display of horizontal grid lines for value axes.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Display of horizontal grids for value axes is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Display of horizontal grid lines for value axes is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarWindowHorizontalGrid`.

BarWindowIndex property

BarWindowIndex

Referenced a configured diagram. Using this attribute, you can assign the values of other attributes to a specific diagram.

Valid values for "BarWindowIndex" are between 0 and "BarWindowCount" minus 1. The "BarWindowCount" attribute specifies the number of configured diagrams.

The "BarWindowIndex" attribute can be assigned dynamic properties using the attribute `BarWindowRepos`. The data type is LONG.

BarWindowName property

Object name - BarWindowName

Specifies the name of the selected diagram window.

The "BarWindowName" attribute can be assigned dynamic properties using the attribute `BarWindowRename`. The data type is STRING.
BarWindowRemove property

**Remove - BarWindowRemove**

Removes the selected diagram window from the list.

The attribute can be assigned dynamic properties using the name `BarWindowRemove`. The data type is `STRING`.

BarWindowRename property

**BarWindowRename**

Renames the diagram window, which is referenced using the "BarWindowIndex" attribute.

The attribute can be assigned dynamic properties using the name `BarWindowRename`. You can use "BarWindowRename" to also assign dynamic properties to the "BarWindowName" attribute. The data type is `STRING`.

BarWindowRepos property

**Up/down - BarWindowRepos**

Changes the order of the diagram windows. "Up" and "Down" move the selected diagram windows up and down in the list.

The order in the list determines the position in the Control. The first diagram window is displayed at the bottommost position, the last diagram window is displayed at the topmost position.

The attribute can be assigned dynamic properties using the name `BarWindowRepos`. The data type is `LONG`.

BarWindowSeriesOverlap property

**Overlap - BarWindowBarOverlap**

Specifies the overlap of various bars of a time instant in % of the bar width.

The attribute can be assigned dynamic properties using the name `BarWindowBarOverlap`. The data type is `LONG`.

BarWindowSpacePortion property

**Proportional area - BarWindowSpacePortion**

Specifies the portion of the selected diagram window for the display in the control.
The attribute can be assigned dynamic properties using the name `BarWindowSpacePortion`. The data type is `LONG`.

**BarWindowVerticalGrid property**

For time axes- `BarWindowVerticalGrid`

Enables the display of vertical grid lines for time axes.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Display of vertical grid lines for time axes is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Display of vertical grid lines for time axes is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `BarWindowVerticalGrid`.

**BarWindowVisible property**

Diagram window - `BarWindowVisible`

The list shows the diagram windows you have created.

Select the diagram windows in the list that you want to display in the control.

Click on an entry in the list to adapt the properties of the bar display and the grid lines.

The attribute can be assigned dynamic properties using the name `BarWindowVisible`. The data type is `BOOLEAN`.

**BarWidth Property**

Description

Defines or returns the width of the bar in pixels.

See also

- 3D Bar (Page 169)
- ScreenItem Object (Page 125)

**BasePicReferenced Property**

Description

TRUE, when the picture assigned in the object status display should be saved. Otherwise, only the associated object reference is saved. Read only access.
See also

Status display (Page 199)
ScreenItem Object (Page 125)

BasePicTransColor Property

Description
Defines or returns which color of the assigned bitmap object (.bmp, .dib) should be set to "transparent". LONG Write/Read Access. The color is only set to "Transparent" if the value of the "BasePicUseTransColor" property is "True".

See also

Status display (Page 199)
ScreenItem Object (Page 125)

BasePicture Property

Description
Returns the basic picture for the object status display. Read-only access. The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated. In this context, the "BasePicReferenced" property defines whether the basic picture should be saved together with the object status display or referenced.

See also

Status display (Page 199)
ScreenItem Object (Page 125)

BasePicUseTransColor Property

Description
TRUE, when the configured color ("BasePicTransColor" property) of the bitmap objects should be set to "transparent". BOOLEAN write-read access.

See also

Status display (Page 199)
ScreenItem Object (Page 125)
BaseScreenName Property

Function

Defines or returns the current basic picture.

STRING (write-read access)

A picture change is executed using the

```
HMIRuntime.BaseScreenName = (<Serverpräfix>::)<Neues Grundbild>
```

command.

When reading out the "BaseScreenName" property, only the picture name without server prefix is returned.

Note

Always enter picture names without the extension "PDL" for reasons of compatibility with future versions.

Example:

The following example executes a picture change to "bild1.pdl":

```
HMIRuntime.BaseScreenName = "bild1"
```

See also

ScreenItem Object (Page 125)
HMIRuntime Object (Page 118)

BaseY Property

Description

Defines or returns the vertical distance of the bottom bar edge to the top edge of the objet field.

See also

3D Bar (Page 169)
ScreenItem Object (Page 125)
BaseX Property

Description
Defines or returns the horizontal distance of the right bar edge to the left edge of the object field in pixels.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Be - Bl

BeginTime Property

Description
WinCC Online Trend Control
The "Index" property references a pair of columns. "BeginTime" defines the start time for displaying this column pair. Write/Read access.

WinCC Online Trend Control
The "Index" property references a trend. "BeginTime" defines the start time for displaying this trend. Whether the information is evaluated is dependent on the TimeRange" and "CommonX" properties.
Use the "yyyy-mm-dd hh:mm:ss" format when creating a dynamic time range.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

BeginValue Property

Description
The "Index" property references a trend. "BeginValue" defines the lower limit of the value range to be displayed for the trend. Whether the information is evaluated is dependent on the "Autorange" and "CommonY" properties.
BeginX Property

Description
Defines or returns the lower limit of the X-axis of a trend referenced with the "Index" property. Whether the information is evaluated is dependent on the "AutorangeX" and "CommonX" properties.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

BeginY Property

Description
Defines or returns the lower limit of the Y-axis of a trend referenced with the "Index" property. Whether the information is evaluated is dependent on the "AutorangeY" and "CommonY" properties.

See also
ScreenItem Object (Page 125)
WinCC Function Trend Control (before WinCC V7) (Page 288)

BevelColorDown Property

Description
Defines the color of the following border sections in the case of 3D representation of the borders:
- with depressed bevel ("BevelStyle" = 1): top and left bevel section
- with raised bevel ("BevelStyle" = 2): bottom and right bevel section

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)
BevelColorUp Property

Description
Defines the color of the following border sections in the case of 3D representation of the borders:
- with depressed bevel ("BevelStyle" = 1): bottom and right bevel section
- with raised bevel ("BevelStyle" = 2): top and left bevel section

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

BevelInner Property

Description
Defines or returns the appearance of the inner part of the object bevel. Write/Read access.
- 0: inner part not available
- 1: "depressed" appearance
- 2: "raised" appearance
- 3: uniform gray border
- 4 or higher: uniformly colored order, border color = background color

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

BevelOuter Property

Description
Defines or returns the appearance of the outer part of the object bevel. Write/Read access.
- 0: inner part not available
- 1: "depressed" appearance
- 2: "raised" appearance
- 3: uniform gray border
- 4 or higher: uniformly colored order, border color = background color
See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

BevelWidth Property

Description
Defines or returns the border width for the inner part of the border (inner bevel) and for the outer border part (outer bevel) in pixels. Write/Read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

BitNumber Property

Description
Defines or returns the bit whose status must change in order to trigger a change of value. The tag used must be of the type BYTE, WORD or DWORD.

See also

Text list (Page 196)
ScreenItem Object (Page 125)

BlinkColor Property

Description
Defines the color of the icon in the flash picture. LONG write-read access.

See also

HMI Symbol Library (Page 239)
ScreenItem Object (Page 125)
BlinkMode property

Flash mode (BlinkMode)

Specifies the flash mode of the icon in runtime.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No flashing</td>
<td>The icon does not flash.</td>
</tr>
<tr>
<td>1</td>
<td>Hidden</td>
<td>The icon flashes in the background color.</td>
</tr>
<tr>
<td>2</td>
<td>Shadow</td>
<td>The icon flashes with shading in the foreground color.</td>
</tr>
<tr>
<td>3</td>
<td>Solid</td>
<td>The icon flashes in the foreground color.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name BlinkMode. The data type is LONG.

BlinkSpeed property

Flash rate (BlinkSpeed)

Specifies the length of the icon flash interval in Runtime.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>Fast</td>
<td>Flash interval of 250 ms.</td>
</tr>
<tr>
<td>500</td>
<td>Medium</td>
<td>Flash interval of 500 ms.</td>
</tr>
<tr>
<td>1000</td>
<td>Slow</td>
<td>Flash interval of 1000 ms.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name BlinkSpeed. You can also use other values. The data type is LONG.

BlockAlign property

Block alignment - BlockAlign

Defines the mode of aligning the caption of blocks in column headers.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The block caption is left justified.</td>
</tr>
<tr>
<td>1</td>
<td>centered</td>
<td>The block caption is aligned to center.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The block caption is right justified.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name BlockAlign. The data type is LONG.
BlockAutoPrecisions property

Decimal places automatic - BlockAutoPrecisions

Enables automatic setting of the decimal precision.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The decimal precision is defined automatically. The value in the &quot;Decimal places&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value in the &quot;Decimal places&quot; field is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name BlockAutoPrecisions. The data type is BOOLEAN.

BlockCaption property

Caption - BlockCaption

Defines the caption of the column header in the control for the selected message block.

The caption is active in all Runtime languages.

The attribute can be assigned dynamic properties by means of the name BlockCaption. The data type is STRING.

BlockCount property

BlockCount

Specifies the number of blocks to be made available as columns for the control.

The attribute can be assigned dynamic properties by means of the name BlockCount. The data type is LONG.

BlockDateFormat property

Date format - BlockDateFormat

Defines the date format for visualization.

The following date formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The date format is set automatically.</td>
</tr>
<tr>
<td>dd.MM.yy</td>
<td>Day.Month.Year, e.g. 24.12.07.</td>
</tr>
<tr>
<td>dd.MM.yyyy</td>
<td>Day.Month.Year, e.g. 24.12.2007.</td>
</tr>
<tr>
<td>dd/MM/yy</td>
<td>Day/Month/Year, e.g. 24/12/07.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/2007.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name **BlockDateFormat**. The data type is STRING.

### BlockExponentialFormat property

**Exponential notation - BlockExponentialFormat**

Specifies exponential notation for the display of values of a selected block.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The values are displayed with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The values are displayed with decimal notation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **BlockExponentialFormat**. The data type is BOOLEAN.

### BlockHideText property

**Content as text - BlockHideText**

Enables the textual display of the content of a selected block.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **BlockHideText**. The data type is BOOLEAN.

### BlockHideTitleText property

**Title as text - BlockHideTitleText**

Enables the display of the header of a selected block in text format.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **BlockHideTitleText**. The data type is BOOLEAN.
BlockId property

**BlockId**

Default assignment of the ID number and of the block in WinCC RulerControl:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No block</td>
</tr>
<tr>
<td>1</td>
<td>Name</td>
</tr>
<tr>
<td>2</td>
<td>Index</td>
</tr>
<tr>
<td>3</td>
<td>Designation</td>
</tr>
<tr>
<td>4</td>
<td>Display</td>
</tr>
<tr>
<td>5</td>
<td>Tag name Y</td>
</tr>
<tr>
<td>6</td>
<td>Tag name X</td>
</tr>
<tr>
<td>7</td>
<td>Y value</td>
</tr>
<tr>
<td>8</td>
<td>X value/time stamp</td>
</tr>
<tr>
<td>9</td>
<td>Y value (LL)</td>
</tr>
<tr>
<td>10</td>
<td>Time stamp (LL)</td>
</tr>
<tr>
<td>11</td>
<td>Y value (UL)</td>
</tr>
<tr>
<td>12</td>
<td>Time stamp (UL)</td>
</tr>
<tr>
<td>13</td>
<td>Minimum</td>
</tr>
<tr>
<td>14</td>
<td>Minimum - Time stamp</td>
</tr>
<tr>
<td>15</td>
<td>Maximum</td>
</tr>
<tr>
<td>16</td>
<td>Maximum - Time stamp</td>
</tr>
<tr>
<td>17</td>
<td>Average</td>
</tr>
<tr>
<td>18</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>19</td>
<td>Integral</td>
</tr>
<tr>
<td>20</td>
<td>Weighted mean value</td>
</tr>
<tr>
<td>21</td>
<td>Duration</td>
</tr>
<tr>
<td>22</td>
<td>Number of values</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **BlockId**. The data type is LONG.

BlockIndex property

**BlockIndex**

References a block. Using this attribute you can assign the values of other attributes to a specific block.

Values between 0 and "BlockCount" minus 1 are valid for "BlockIndex". Attribute "BlockCount" defines the number of available blocks.

The attribute can be assigned dynamic properties by means of the name **BlockIndex**. The data type is LONG.
BlockLength property

Length in characters - BlockLength

Specifies the column width for a selected block.

The attribute can be assigned dynamic properties by means of the name BlockLength. The data type is LONG.

BlockName property

Object name - BlockName

Displays the name of the block selected. You cannot edit this name.

The attribute can be assigned dynamic properties by means of the name BlockName. The data type is STRING.

BlockPrecisions property

Decimal places - BlockPrecisions

Specifies the number of decimal places of the values in the selected column. You can only enter the value if the "Automatic" option is disabled.

The attribute can be assigned dynamic properties by means of the name BlockPrecisions. The data type is SHORT.

BlockShowDate property

Display date - BlockShowDate

Specifies if the "Time" block is displayed with time and date in a field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The date and time are displayed. The date format is defined in the &quot;Date format&quot; field.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time is displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name BlockShowDate. The data type is BOOLEAN.

BlockShowIcon property

Content as icon - BlockShowIcon
Enables the display of the content of a selected block as icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is visualized as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is not visualized as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `BlockShowIcon`. The data type is BOOLEAN.

**BlockShowTitleIcon property**

*Title as icon - BlockShowTitleIcon*

Enables the display of the header of a selected block as icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is displayed as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is not displayed as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `BlockShowTitleIcon`. The data type is BOOLEAN.

**BlockTimeFormat property**

*Time format - BlockTimeFormat*

Defines the time format to be used for visualization.

The following time formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time format is set automatically.</td>
</tr>
<tr>
<td>HH:mm:ss.ms</td>
<td>Hours:Minutes:Seconds, e.g. 15:35:44.240.</td>
</tr>
<tr>
<td>hh:mm:ss tt</td>
<td>Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.</td>
</tr>
<tr>
<td>hh:mm:ss.ms tt</td>
<td>Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `BlockTimeFormat`. The data type is STRING.

**BlockUseSourceFormat property**

*Use source format - BlockUseSourceFormat*
Specifies that the format is inherited from the interconnected control. Here the size of the control, the zoom factor and the value range are taken into consideration to display the optimal number of decimal places.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The formats are derived from the interconnected control.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The formats configured in Ruler Control are used, for example, the display of a precisely specified number of decimal places.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `BlockUseSourceFormat`. The data type is BOOLEAN.

**Bo - Bu**

**BorderBackColor Property**

**Description**

Defines or returns the background color of the line for the object. LONG write-read access. The background color is only visible with the property setting "BorderWidth" > 0.

**See also**

ScreenItem Object (Page 125)

**BorderColor Property**

**Description**

Defines or returns the line color for the object. LONG write-read access.

**See also**

ScreenItem Object (Page 125)

**BorderColor property**

**Border color - BorderColor**

Specifies the border color. The button opens the "Color selection" dialog. The attribute can be assigned dynamic properties by means of the name `BorderColor`. The data type is LONG.
BorderColorBottom Property

Description
Defines or returns the border color for the bottom/right part of the object. LONG write-read access.

See also
ScreenItem Object (Page 125)
Button (Page 201)
Round Button (Page 209)

BorderColorTop Property

Description
Defines or returns the border color for the top/left part of the object. LONG write-read access.

See also
Button (Page 201)
Round Button (Page 209)
ScreenItem Object (Page 125)

BorderEndStyle Property

Description
Defines or returns the line end style of the object. LONG write-read access.

See also
Polyline (Page 158)
Line (Page 154)
ScreenItem Object (Page 125)

BorderFlashColorOff Property

Description
Defines or returns the color of the object lines for the flashing status "Off". LONG write-read access.
See also

ScreenItem Object (Page 125)

BorderFlashColorOn Property

Description
Defines or returns the color of the object lines for the flashing status "On". LONG write-read access.

See also
ScreenItem Object (Page 125)

BorderStyle Property

Description
Defines or returns the line style for the object. Value range from 0 to 4.
0 = solid line
1 = dashed line
2 = dotted line
3 = dash-dotted line
4 = dash-dot-dot line

See also
ScreenItem Object (Page 125)

BorderWidth Property

Description
Defines or returns the line weight (in pixels) for the object.

WinCC Gauge Control:
Defines or returns the width of the middle border part in pixels.
The object border is composed of three parts. The middle part of the object border is described by the "BorderWidth" property.
The color of the middle border part is in the background color.
See also

ScreenItem Object (Page 125)

BorderWidth property

Border width - BorderWidth

Specifies the line weight of the border in pixels.

The attribute can be assigned dynamic properties by means of the name BorderWidth. The data type is LONG.

BottomConnectedConnectionPointIndex Property

Description

Specifies or sets the index number of the bottom connecting point.

LONG write-read access.

See also

Connector (Page 167)

ScreenItem Object (Page 125)

BottomConnectedObjectName Property

Description

Specifies or sets the object name of the object which is docked on at the bottom connecting point.

LONG write-read access.

See also

Connector (Page 167)

ScreenItem Object (Page 125)

BoxAlignment Property

Description

TRUE, when the fields are arranged aligned to the right. BOOLEAN write-read access.
See also
Radio box (Page 207)
Check box (Page 204)
ScreenItem Object (Page 125)

BoxCount Property

Description
Defines or returns the number of fields. Value range from 0 to 31.

See also
Radio box (Page 207)
Check box (Page 204)
ScreenItem Object (Page 125)

BoxType Property

Description
Defines or returns the field type. Value range from 0 to 2:

- 0: Edition
- 1: Input
- 2: I/O field

See also
Text list (Page 196)
I/O Field (Page 184)
ScreenItem Object (Page 125)

ButtonColor Property

Description
Defines or returns the color of the slider. LONG write-read access.
See also

Slider (Page 212)
ScreenItem Object (Page 125)

ButtonCommand Property

Description

Upon changing a value of "ButtonCommand", a message is issued to the WinCC Alarm Control in order to adapt the display in the message window.

Value (hex); value (dec); Retrieved Function:

- 0x00000001; 1; Message list
- 0x00000002; 2; Short-term archive list
- 0x00000004; 4; Long-term archive list
- 0x00200000; 2097152; Lock list
- 0x00000008; 8; Acknowledge central signaling device
- 0x00000010; 16; Single Acknowledgment
- 0x00000020; 32; Group Acknowledge
- 0x00000040; 64; Autoscroll
- 0x00000080; 128; Selection Dialog
- 0x00000100; 256; Lock Dialog
- 0x00000200; 512; Print Message Log
- 0x00000800; 2048; Emergency Acknowledgment
- 0x00001000; 4096; First Message
- 0x00002000; 8192; Last Message
- 0x00004000; 16384; Next Message
- 0x00008000; 32768; Previous Message
- 0x00010000; 65536; Infotext Dialog
- 0x00020000; 131072; Comments Dialog
- 0x00040000; 262144; Loop in Alarm
- 0x00100000; 1048576; Print current view
- 0x00400000; 4194304; Lock/unlock message
- 0x00800000; 8388608; Sorting Dialog
- 0x01000000; 16777216; Time base dialog
- 0x02000000; 33554432; Hit list
- 0x04000000; 67108864; List of messages to be hidden
• 0x08000000; 134217728; Show/hide message
• 0x10000000; 268435456; Display option dialog

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Button1MessageClasses property

Message Types for Button 1 (Button1MessageClasses)
Define one or more message events for displaying the first button in the group display. This is done by entering the numbers of the bits in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, separate the numbers with a comma. The sequence of the assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

One event can be displayed simultaneously in more than one button.

The "Message Types for Button 1" attribute can be assigned dynamic properties with the name "Button1MessageClasses".

Button2MessageClasses property

Message Types for Button 2 (Button2MessageClasses)
For displaying both buttons, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, separate the numbers with a comma. The sequence of the assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 2" attribute can be assigned dynamic properties with the name "Button2MessageClasses".

Button3MessageClasses property

Message Types for Button 3 (Button3MessageClasses)
For displaying the third button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, separate the numbers with a comma. The sequence of the assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.
The "Message Types for Button 3" attribute can be assigned dynamic properties with the name "Button3MessageClasses".

**Button4MessageClasses property**

**Message Types for Button 4 (Button4MessageClasses)**

For displaying the fourth button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, separate the numbers with a comma. The sequence of the assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 4" attribute can be assigned dynamic properties with the name "Button4MessageClasses".

**Button5MessageClasses property**

**Message Types for Button 5 (Button5MessageClasses)**

For displaying the fifth button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, separate the numbers with a comma. The sequence of the assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 5" attribute can be assigned dynamic properties with the name "Button5MessageClasses".

**Button6MessageClasses property**

**Message Types for Button 6 (Button6MessageClasses)**

For displaying the sixth button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, delimit the numbers with a comma. The order of assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 6" attribute can be assigned dynamic properties with the name "Button6MessageClasses".
**Button7MessageClasses property**

**Message Types for Button 7 (Button7MessageClasses)**

For displaying the seventh button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, delimit the numbers with a comma. The order of assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 7" attribute can be assigned dynamic properties with the name "Button7MessageClasses".

**Button8MessageClasses property**

**Message Types for Button 8 (Button8MessageClasses)**

For displaying the eighth button, define one or more message events in the group display. This is done by entering the number of the bit in the collect value. The display of the message events is configured in the "Message Types" property group.

If you want to assign several message events, delimit the numbers with a comma. The order of assignment defines the priority. If there are more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

The "Message Types for Button 8" attribute can be assigned dynamic properties with the name "Button8MessageClasses".

**Button1Width Property**

**Description**

Defines or returns the width of the Button 1 in pixels. When the SameSize property is set to TRUE, all the buttons are specified the same width.

See also

- Group Display (Page 194)
- ScreenItem Object (Page 125)

**Button2Width Property**

**Description**

Defines or returns the width of the Button 2 in pixels. When the SameSize property is set to TRUE, all the buttons are specified the same width.
See also

Group Display (Page 194)
ScreenItem Object (Page 125)

Button3Width Property

Description
Defines or returns the width of the Button 3 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

See also

ScreenItem Object (Page 125)
Group Display (Page 194)

Button4Width Property

Description
Defines or returns the width of the Button 4 in pixels.
When the SameSize property is set to TRUE, all the buttons are specified the same width.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

1.14.4.4 C

Ca - Cl

Caption Property

Description
Application and picture windows
TRUE, when the application or picture window has a title bar in Runtime. Read only access.
The Caption property must be set to TRUE when the application or picture window should have Maximize and Close buttons.
Controls before WinCC V7
Defines or returns the text to be displayed on the label on the button or in the title bar (Online Trend Control and Online Table Control). Write/Read access.

See also
Controls (Page 218)
Picture Window (Page 179)
Application Window (Page 173)
ScreenItem Object (Page 125)

Caption property

Text - Caption
Defines the text of the window caption.
The attribute can be assigned dynamic properties by means of the name Caption. The data type is STRING.

CaptionColor Property

Description
Defines or returns the color of the element labeling. LONG write-read access.

See also
ScreenItem Object (Page 125)
WinCC Gauge Control (Page 254)

CaptionFont Property

Description
Returns the values for font, font style and font size as well as the "Underline" and "Strikethrough" effects for the element labeling. Read only access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)
CaptionOffset Property

Description
Defines or returns the distance of the element labeling in relation to the top edge of the object. The element labeling can only be positioned along the vertical diameter of the graduated scale disk. The value of the attribute is related to the height of the object and is measured from the top edge of the object to the base of the text. Write/Read access.

The value range is 0 to 1:

0: The base of the text is at the top limit of the object. The text is no longer visible because it is outside the object.
1: The base of the text is at the bottom limit of the object.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

CaptionText Property

Description
Defines or returns the window title which is displayed in Runtime. The Caption property must be set to TRUE.

See also
Picture Window (Page 179)
ScreenItem Object (Page 125)

CellCount property

CellCount
Specifies the number of cells of the Row object of a Table Control. The number of cells corresponds to the number of columns.

CellCut property (before WinCC V7)

Description
TRUE, when the content of the cells in a message line should be cut if the column width is too small. BOOLEAN write-read access.
See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

CellCut property

Shorten contents - CellCut
Shortens cell contents if the cell width is insufficient.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables shortening of cell contents.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables shortening of cell contents.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `CellCut`. The data type is BOOLEAN.

CellSpaceBottom property

CellSpaceBottom
Defines the bottom margin of the table cells.

The attribute can be assigned dynamic properties by means of the name `CellSpaceBottom`. The data type is LONG.

CellSpaceLeft property

CellSpaceLeft
Defines the left indent of the table cells.

The attribute can be assigned dynamic properties by means of the name `CellSpaceLeft`. The data type is LONG.

CellSpaceRight property

CellSpaceRight
Defines the right indent of the table cells.

The attribute can be assigned dynamic properties by means of the name `CellSpaceRight`. The data type is LONG.
CellSpaceTop property

CellSpaceTop

Defines the top margin of the table cells. The attribute can be assigned dynamic properties by means of the name `CellSpaceTop`. The data type is LONG.

CellText property

CellText

Returns the contents of a cell as a string. The cell is determined from the column number of the row object, e.g. `CellText(3)` for the third column. Numbering runs from "1" to "CellCount".

CenterColor Property

Description

Defines or returns the color of the circular center of the scale (cover of the pointer axis). LONG write-read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

CenterScale Property

Description

Defines or returns the diameter of the circular center of the scale (cover of the pointer axis) in relation to the smaller value of the geometric width and height attributes. Write/Read access. The value range is 0.03 to 1:

1: The diameter corresponds to the smaller value of the "Width" or "Height" geometric values.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)
CheckAlarmHigh Property

Description
TRUE, when the "AlarmHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "AlarmHigh", "ColorAlarmHigh" and "TypeAlarmHigh" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

CheckAlarmLow Property

Description
TRUE, when the "AlarmLow" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "AlarmLow", "ColorAlarmLow" and "TypeAlarmLow" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

CheckLimitHigh4 Property

Description
TRUE, when the "Reserve 4" upper limit value should be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitHigh4", "ColorLimitHigh4" and "TypeLimitHigh4" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
CheckLimitHigh5 Property

Description
TRUE, when the "Reserve 5" upper limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitHigh5", "ColorLimitHigh5" and "TypeLimitHigh5" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

CheckLimitLow4 Property

Description
TRUE, when the "Reserve 4" lower limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitLow4", "ColorLimitLow4" and "TypeLimitLow4" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

CheckLimitLow5 Property

Description
TRUE, when the "Reserve 5" lower limit value should be monitored. BOOLEAN write/read access.
The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "LimitLow5", "ColorLimitLow5" and "TypeLimitLow5" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
CheckToleranceHigh Property

Description

TRUE, when the "ToleranceHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "ToleranceHigh", "ColorToleranceHigh" and "TypeToleranceHigh" properties.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

CheckToleranceLow Property

Description

TRUE, when the "ToleranceLow" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "ToleranceLow", "ColorToleranceLow" and "TypeToleranceLow" properties.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

CheckWarningHigh Property

Description

TRUE, when the "WarningHigh" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "WarningHigh", "ColorWarningHigh" and "TypeWarningHigh" properties.

See also

Bar (Page 174)
ScreenItem Object (Page 125)
CheckWarningLow Property

Description
TRUE, when the "WarningLow" limit value is to be monitored. BOOLEAN write/read access. The limit value, the display on reaching the limit value and the type of evaluation are defined by means of the "WarningLow", "ColorWarningLow" and "TypeWarningLow" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ClearOnError Property

Description
TRUE, when the field entry is automatically deleted in the case of invalid input. BOOLEAN write-read access.

See also
I/O Field (Page 184)
ScreenItem Object (Page 125)

ClearOnNew Property

Description
TRUE, when the field entry is deleted as soon as the I/O field has the focus. BOOLEAN write-read access.

See also
I/O Field (Page 184)
ScreenItem Object (Page 125)

Closeable property (before WinCC V7)

Description
TRUE, when the window can be closed in Runtime. BOOLEAN write-read access.
Closeable property

Closeable

Defines whether the control can be closed in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The control can be closed in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The control cannot be closed in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `Closeable`. The data type is BOOLEAN.

CloseButton Property

Description

TRUE, when the window is provided with a "Close" button. Read only access.

CoarseGrid Property

Description

TRUE when the value axis is scaled by long tick marks. The distance between two long tick marks can be changed using the "CoarseGridValue" property. BOOLEAN write-read access.
See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

CoarseGridX Property

Description
TRUE, when the X-axis graduation is scaled by long tick marks. The distance between two long tick marks can be changed using the "CoarseGridValueX" property. BOOLEAN write-read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CoarseGridY Property

Description
TRUE, when the Y-axis graduation is scaled by long tick marks. The distance between two long tick marks can be changed using the "CoarseGridValueY" property. BOOLEAN write-read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CoarseGridValue Property

Description
Defines the distance between two long tick marks in the scale. Whether the information is evaluated is dependent on the value of the "CoarseGrid" property.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)
CoarseGridValueX Property

Description
Defines or returns the distance between two long tick marks on the graduation scale of the X-axis. Whether the information is evaluated is dependent on the value of the "CoarseGridX" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CoarseGridValueY Property

Description
Defines or returns the distance between two long tick marks on the graduation scale of the Y-axis. Whether the information is evaluated is dependent on the value of the "CoarseGridY" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CollectValue Property

Description
Contains the respective status of the active message class in Runtime as the start value. LONG write/read access.
The value can be determined from the group display of hierarchically subordinate pictures by making it dynamic using a tag.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)
ColMove Property

Description
TRUE, when the arrangement of columns can be changed. BOOLEAN write-read access.

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Color Property

Description
The "Index" property references a column pair or a trend. "Color" defines the color of the font in the column or the trend. LONG write-read access. The color is defined as an RGB value.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ColorAlarmHigh Property

Description
Defines or returns the bar color for the "AlarmHigh" limit value. LONG write/read access. The "CheckAlarmHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
ColorAlarmLow Property

Description
Defines or returns the bar color for the "AlarmLow" limit value. LONG write/read access. The "CheckAlarmLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorBottom Property

Description
Defines or returns the color for the bottom/right stop of the slider object. LONG write-read access.

See also
Slider (Page 212)
ScreenItem Object (Page 125)

ColorChangeType Property

Description
TRUE, if the change of color should occur segment by segment in the case of a color change (e.g. on reaching a limit value). If set to FALSE, it defines the change of color for the entire bar. BOOLEAN write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
ColorLimitHigh4 Property

Description
Defines or returns the color for the "Reserve 4" upper limit value. LONG write/read access. The "CheckLimitHigh4" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorLimitHigh5 Property

Description
Defines or returns the color for the "Reserve 5" upper limit value. LONG write/read access. The "CheckLimitHigh5" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorLimitLow4 Property

Description
Defines or returns the color for the "Reserve 4" lower limit value. LONG write/read access. The "CheckLimitLow4" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
ColorLimitLow5 Property

Description
Defines or returns the color for the "Reserve 5" lower limit value. LONG write/read access. The "CheckLimitLow5" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorToleranceHigh Property

Description
Defines or returns the color for the "ToleranceHigh" upper limit value. LONG write/read access. The "CheckToleranceHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorToleranceLow Property

Description
Defines or returns the color for the "ToleranceLow" lower limit value. LONG write/read access. The "CheckToleranceLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ColorTop Property

Description
Defines or returns the color for the top/left stop of the slider object. LONG write-read access.
See also

Slider (Page 212)
ScreenItem Object (Page 125)

ColorWarningHigh Property

Description
Defines or returns the color for the "WarningHigh" upper limit value. LONG write/read access. The "CheckWarningHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

ColorWarningLow Property

Description
Defines or returns the color for the "WarningLow" lower limit value. LONG write/read access. The "CheckWarningLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

ColTitle Property

Description
TRUE, when the columns in the message window should have a title bar. BOOLEAN write-read access.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)
ColumnAdd property

**Apply - ColumnAdd**

Copies the selected column from the list of existing columns to the list of selected columns. The attribute can be assigned dynamic properties by means of the name ColumnAdd. The data type is STRING.

ColumnAlias property

**ColumnAlias**

Defines the alias specified in the user archive for the column name. The attribute can be assigned dynamic properties by means of the name ColumnAlias. The data type is STRING.

ColumnAlign property

**Alignment - ColumnAlign**

Specifies the mode of alignment of a selected column. The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The selected column is aligned left.</td>
</tr>
<tr>
<td>1</td>
<td>centered</td>
<td>The selected column is aligned to center.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The selected column is aligned right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ColumnAlign. The data type is LONG.

ColumnAutoPrecisions property

**Decimal places automatic - ColumnAutoPrecisions**

Enables automatic setting of the decimal precision.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The decimal precision is defined automatically. The value in the &quot;Decimal places&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value in the &quot;Decimal places&quot; field is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ColumnAutoPrecisions. The data type is BOOLEAN.
**ColumnCaption property**

**Caption - ColumnCaption**

Sets the caption for a selected column.

The attribute can be assigned dynamic properties by means of the name `ColumnCaption`. The data type is STRING.

**ColumnCount property**

**ColumnCount**

Defines the number of columns configured.

The attribute can be assigned dynamic properties by means of the name `ColumnCount`. The data type is LONG.

**ColumnDateFormat property**

**Date format - ColumnDateFormat**

Defines the date format for visualization.

The following date formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The date format is set automatically.</td>
</tr>
<tr>
<td>dd.MM.yy</td>
<td>Day.Month.Year, e.g. 24.12.07.</td>
</tr>
<tr>
<td>dd.MM.yyyy</td>
<td>Day.Month.Year, e.g. 24.12.2007.</td>
</tr>
<tr>
<td>dd/MM/yy</td>
<td>Day/Month/Year, e.g. 24/12/07.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/2007.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnDateFormat`. The data type is STRING.

**ColumnDMVarName property**

**ColumnDMVarName**

Defines the name of the tag you assigned to the column in the user archive.

The attribute can be assigned dynamic properties by means of the name `ColumnDMVarName`. The data type is STRING.
ColumnExponentialFormat property

Exponential notation - ColumnExponentialFormat

Sets exponential notation for the display of values of a selected column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The values are displayed with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The values are displayed with decimal notation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ColumnExponentialFormat. The data type is BOOLEAN.

ColumnFlagNotNull property

ColumnFlagNotNull

Specifies whether the user archive field assigned to the column must have a value.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>The column must have a value.</td>
</tr>
<tr>
<td>No</td>
<td>The column can have a value.</td>
</tr>
</tbody>
</table>

The attribute cannot be dynamized.

ColumnFlagUnique property

ColumnFlagUnique

Specifies whether the user archive field assigned to the column must have a unique value. Values in this column must not be redundant.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The column must have a unique value.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The column must not have a unique value.</td>
</tr>
</tbody>
</table>

The attribute cannot be dynamized.

ColumnHideText property

Content as text - ColumnHideText

Defines textual display of the contents of a selected column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ColumnHideText. The data type is BOOLEAN.
**ColumnHideTitleText property**

**Text header - ColumnHideTitleText**

Sets textual display of the header of a selected column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnHideTitleText`. The data type is BOOLEAN.

---

**ColumnId property**

**ColumnId**

Default assignment of the ID number and column in the WinCC control:

The attribute can be assigned dynamic properties by means of the name `ColumnId`. The data type is LONG.

---

**ColumnIndex property**

**ColumnIndex**

References a control column. Using this attribute you can assign the values of other properties to a specific column.

Values between 0 and "ColumnCount" minus 1 are valid for "ColumnIndex"; the attribute "ColumnCount" defines the number of available columns.

The "ColumnIndex" attribute can be assigned dynamic properties by means of attribute `ColumnIndex`. The data type is LONG.

---

**ColumnInInspectorView property**

**ColumnInInspectorView**

Specifies whether the diagnostic data in the detail view are displayed in columns instead of rows.

The attribute can be assigned dynamic properties using the name `ColumnInInspectorView`. The data type is BOOLEAN.
ColumnLeadingZeros property

With leading zeros - ColumnLeadingZeros

Enables the display of values with leading zeros for the column selected. Use "Number of digits" or "ColumnLeadingZeros" to specify the number of leading zeros. The maximum number is "11". No leading zeros are displayed with the value "0". The "With leading zeros" option is deactivated.

The attribute can be assigned dynamic properties by means of the name ColumnLeadingZeros. The data type is LONG.

ColumnLength property

Length in Characters - ColumnLength

Specifies the width of a selected column.

The attribute can be assigned dynamic properties by means of the name ColumnLength. The data type is LONG.

ColumnMaxValue property

ColumnMaxValue

Defines the maximum column value specified in the user archive.

The attribute can be assigned dynamic properties by means of the name ColumnMaxValue. The data type is STRING.

ColumnMinValue property

ColumnMinValue

Defines the minimum column value specified in the user archive.

The attribute can be assigned dynamic properties by means of the name ColumnMinValue. The data type is STRING.

ColumnName property

ColumnName

Defines the name of the column which is referenced by means of "ColumnIndex" attribute.

The attribute can be assigned dynamic properties by means of the name ColumnName. The data type is STRING.
ColumnPosition property

ColumnPosition

Displays the field position defined in the user archive.

The attribute can be assigned dynamic properties by means of the name `ColumnPosition`. The data type is LONG.

ColumnPrecisions property

Decimal places - ColumnPrecisions

Specifies the number of decimal places of the values in the selected column. You can only enter the value if the "Automatic" option is disabled.

The attribute can be assigned dynamic properties by means of the name `ColumnPrecisions`. The data type is SHORT.

ColumnReadAccess property

ColumnReadAccess

Defines authorizations for read access to the column as specified in the user archive. The number corresponds with the number assigned to the authorization in the "User Administrator" editor.

The attribute cannot be dynamized.

ColumnReadonly property

Write protected - ColumnReadonly

Sets the write protection of a selected column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>This column is write protected.</td>
</tr>
<tr>
<td>FALSE</td>
<td>This column is not write protected. You can edit the column values in Runtime by activating the &quot;Change&quot; option in the General&quot; tab.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnReadonly`. The data type is BOOLEAN.

ColumnRemove property

Remove - ColumnRemove

Cuts selected columns from the list of selected columns and pastes these to the list of available columns.
The attribute can be assigned dynamic properties by means of the name `ColumnRemove`. The data type is STRING.

**ColumnRepos property**

**Up/Down - ColumnRepos**

Changes the order of columns. "Up" and "Down" move the column selected up or down in the list. This moves the column towards the front or towards the back.

The attribute can be assigned dynamic properties by means of the name `ColumnRepos`. The data type is LONG.

**ColumnResize property**

**Width can be resized - ColumnResize**

Enables changes to the width of columns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>You can change the width of the columns.</td>
</tr>
<tr>
<td>FALSE</td>
<td>You cannot change the width of the columns.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnResize`. The data type is BOOLEAN.

**ColumnScrollbar properties**

**Column scroll bars - ColumnScrollbar**

Enables the display of column scroll bars.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
<td>Column scroll bars are not displayed.</td>
</tr>
<tr>
<td>1</td>
<td>as required</td>
<td>Column scroll bars are displayed if vertical space requirements of the con‐</td>
</tr>
<tr>
<td></td>
<td></td>
<td>trol are greater than the actually available display area.</td>
</tr>
<tr>
<td>2</td>
<td>always</td>
<td>Column scroll bars are always displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnScrollbar`. The data type is LONG.
**ColumnShowDate property**

*Display date - ColumnShowDate*

Specifies if the "Time" block is displayed with time and date in a field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The date and time are displayed. The date format is defined in the &quot;Date format&quot; field.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time is displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnShowDate`. The data type is BOOLEAN.

**ColumnShowIcon property**

*Content as icon - ColumnShowIcon*

Enables the display the contents of a selected column by means of icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is visualized as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is not visualized as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnShowIcon`. The data type is BOOLEAN.

**ColumnShowTitleIcon property**

*Header as icon - ColumnShowTitleIcon*

Specifies the display of the header of a selected column by means of icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is displayed as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is not displayed as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnShowTitleIcon`. The data type is BOOLEAN.

**ColumnsMoveable property**

Specifies whether the user can move the columns of the control in runtime.

The attribute can be assigned dynamic properties using the name `ColumnsMoveable`. The data type is BOOLEAN.
ColumnSort property

ColumnSort

Defines the sorting order of the user archive column referenced in the "ColumnIndex" attribute.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No sorting</td>
</tr>
<tr>
<td>1</td>
<td>Ascending</td>
<td>Ascending order, starting at the lowest value.</td>
</tr>
<tr>
<td>2</td>
<td>Descending</td>
<td>Descending order, starting at the highest value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ColumnSort. The data type is LONG.

ColumnSortIndex property

ColumnSortIndex

Defines the sorting order of the column referenced in "ColumnIndex". The sorting criterion is removed from "ColumnSort" if you set a "0" value.

The attribute can be assigned dynamic properties by means of the name ColumnSortIndex. The data type is LONG.

ColumnStartValue property

ColumnStartValue

Defines the column start value specified in the user archive.

The attribute can be assigned dynamic properties by means of the name ColumnStartValue. The data type is STRING.

ColumnStringLength property

ColumnStringLength

Displays the string length of the column as defined in the user archive.

The attribute can be assigned dynamic properties by means of the name ColumnStringLength. The data type is LONG.
**ColumnTimeFormat property**

**Time format - ColumnTimeFormat**

Defines the time format to be used for visualization.

The following time formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time format is set automatically.</td>
</tr>
<tr>
<td>HH:mm:ss.ms</td>
<td>Hours:Minutes:Seconds, e.g. 15:35:44.240.</td>
</tr>
<tr>
<td>hh:mm:ss tt</td>
<td>Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.</td>
</tr>
<tr>
<td>hh:mm:ss.ms tt</td>
<td>Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnTimeFormat`. The data type is STRING.

**ColumnTitleAlign property**

**Column title alignment - ColumnTitleAlign**

Specifies the type of column title alignment.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The column titles are left justified.</td>
</tr>
<tr>
<td>1</td>
<td>centered</td>
<td>The column titles are centered.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The column titles are right justified.</td>
</tr>
<tr>
<td>3</td>
<td>Same as table content</td>
<td>The column titles are justified to fit the corresponding column content.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnTitleAlign`. The data type is LONG.

**ColumnTitles property**

**Show column title - ColumnTitles**

Enables the display of the column header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The column header is displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The column header is not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnTitles`. The data type is BOOLEAN.
ColumnType property

**ColumnType**

Indicates the associated view of the selected column in the SysDiagControl.

The attribute can be assigned dynamic properties using the name `ColumnType`. The data type is LONG.

ColumnVisible property

**ColumnVisible**

Enables the display of a column referenced by means of "ColumnIndex" attribute.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The column is displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The column is not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ColumnVisible`. The data type is BOOLEAN.

ColumnWriteAccess property

**ColumnWriteAccess**

Defines authorizations for write access to the column as specified in the user archive. The number corresponds with the number assigned to the authorization in the "User Administrator" editor.

The attribute cannot be dynamized.

ColWidth Property

**Description**

TRUE, when it should be possible to change the widths of the columns in the message window. The width of the columns can only be changed, however, when the "AutoScroll" property is not active. BOOLEAN write-read access.
See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Command Property

Description
TRUE, when updating of the values displayed in the control should be forced.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

Comment property

Description
Reads or sets the Alarm object comment.

See also

Alarms object (list) (Page 110)

CommonTime Property

Description
TRUE, when a common time column is to be used in the table window. BOOLEAN write-read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)
CommonX Property

Description
TRUE, when the trends in the trend window should be displayed with a common X-axis. BOOLEAN write-read access.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CommonY Property

Description
TRUE, when the trends in the trend window should be displayed with a common Y-axis. BOOLEAN write-read access.

See also
ScreenItem Object (Page 125)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)

ComputerName property

Description
Returns the name of the computer on which the alarm object was triggered.
ComputerName (readonly)

See also
Alarms object (list) (Page 110)

Context property

Description
Reads or sets the alarm object server prefix.
See also

Alarms object (list) (Page 110)

ConnectBarWindows property

**Connected diagram window - ConnectBarWindows**

Specifies whether the configured diagram windows are connected. You must have configured several diagram windows for this.

The connected diagram windows have the following properties:

- They can have a common X axis.
- They have a scroll bar.
- The zoom functions for a diagram window affect the connected diagram windows.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>All configured diagram windows are connected.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The diagram windows are displayed separately.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name **ConnectBarWindows**. The data type is BOOLEAN.

ConnectTrendWindows property

**Connect trend windows - ConnectTrendWindows**

Enables the connection of trend windows configured. You must have configured several trend windows.

The connected trend windows have the following properties:

- They can have a common X axis.
- They have a scroll bar.
- They have a ruler.
- The zoom functions for a trend window affect the connected trend windows.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>All trend windows configured are connected.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The trend windows are displayed separately.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **ConnectTrendWindows**. The data type is BOOLEAN.
ContinuousChange Property

Description
Defines the type of transfer of the value defined by the slider ("Position" property) in Runtime:

- FALSE: The value of the "Position" property is transferred when the mouse button is released.
- TRUE: The value of the "Position" property is transferred immediately following a change of the slider position.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

Count Property

Description
Supplies the number of elements in a list.
INTEGER (read-only access).

Example:
The example shows how the number of objects in a DataSet list is output.

'VBS165

The following example adds two tags to the TagSet list and outputs the count properties as Trace.

'VBS177
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "Count: " & group.Count & vbNewLine
Count property

Count

Specifies the number of rows of the row listing object of a table control.

CurrentContext Property

Description

In the case of a picture window, the server from which the picture comes and contains the script is read out.

The "CurrentContext" property can return different results: If, for example, a picture window displaying a server picture is set in a local basic picture, distinction is made between two cases:

- The "CurrentContext" property is used in an action of the picture window picture: The result is the return of the symbolic computer name of the server (Package property) extended by two colons, e.g."WinCCProject_MyComputer::".
- The "CurrentContext" property is used in an action of the basic picture: The result is returned in the form of an empty character string.

See also

HMIRuntime Object (Page 118)
Cursor Property

Description
Controls the appearance of the cursor in Runtime when positioned over an icon.

- 0: The cursor appears as an arrow and does not change when positioned over the icon.
- 1: The cursor appears as a 3D arrow accompanied by a green lightening symbol. In Runtime, this indicates that the object concerned can be operated.

See also
ScreenItem Object (Page 125)
HMI Symbol Library (Page 239)

Cursor property

Mouse pointer (Cursor)
Specifies whether or not to display the mouse pointer on the icon at runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The mouse pointer is shown at runtime if positioned on the icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The mouse pointer is hidden at runtime if positioned on the icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **Cursor**. The data type is BOOLEAN.

CursorControl Property

Description
TRUE, when Alpha Cursor mode is activated, the cursor skips to the next field in the TAB sequence after exiting the field. BOOLEAN write-read access.

To do this, the "CursorMode" property must be set to TRUE.

See also
Text list (Page 196)
I/O Field (Page 184)
ScreenItem Object (Page 125)
CurveForm Property

Description

WinCC Function Trend Control
Defines how the measuring points of a trend referenced by the "Index" property should be connected. Write/Read access.

WinCC Online Trend Control
The "Index" property references a trend. "CurveForm" defines how the measuring points should be connected.

- 0x00000012 Representation of the measuring points.
- 0x00000014 Measuring points are connected linearly.
- 0x00000011 Measuring points are connected via a step curve.
- 0x00000021 The area under the linearly connected trend is filled.
- 0x00000022: The area under the step curve is filled.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

CursorMode Property

Description

When the "CursorMode" is set to "yes", you can show all messages from the short-term archive page by page in the long-term archive list. Use the "CursorModePrefetch" property to determine the number of messages shown per page.

The "Autoscroll" option must be unchecked in order to be able to switch between pages. Write/Read access.

CursorModePrefetch Property

Description

Sets the number of message that you want to display page by page in the long-term archive list out of all messages in the short-term archive.

The "CursorMode" object property must be set to "yes".

Write/Read access.
1.14.4.5 D

Da

DangerColor Property

Description
Defines or returns the color of the danger zone on the scale. LONG write-read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

Danger Property

Description
Defines or returns the beginning of the "danger zone". The zone stretches from the "danger" value to the end of the scale. Write/Read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

DataFormat Property

Description
Returns the data type of the I/O field object. Read only access.
Value range from 0 to 3.
0: Binary
1: Decimal
2: String
3: Hexadecimal
See also

I/O Field (Page 184)
ScreenItem Object (Page 125)

**DataIndex Property**

**Description**

Returns the current index of the data of the current trend.

**Note**

The property is only supported for the controls prior to WinCC V7.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**DataLogs Property**

**Description**

Returns an object of type "DataLogs".

DataLogs (read-only)

See also

DataLogs Object (Page 114)
HMIRuntime Object (Page 118)

**DataSet Property**

**Description**

Returns an object of type "DataSet".

DataSet (read-only)
See also

DataSet Object (List) (Page 116)
HMIRuntime Object (Page 118)

DataX Property

Description
Inserts a single data record and must be set before calling "InsertData".

Note
The property is only supported for the controls prior to WinCC V7.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

DataXY Property

Description
Inserts several data records as an array with pairs of values and must be set before calling "InsertData".
The data in the array is assumed when "DataX" is of the VT_EMPTY type. Otherwise, the "InsertData" attribute used the single value pair resulting from “DataX” and “DataY”.

Note
The property is only supported for the controls prior to WinCC V7.

See also

Example: Calling Methods of an ActiveX Control (Page 841)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
DataY Property

Description

Inserts a single data record and must be set before calling "InsertData".

Note

The property is only supported for the controls prior to WinCC V7.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

De - Do

DefaultMsgFilterSQL property

DefaultMsgFilterSQL

Defines an SQL statement for a fixed selection of messages.

The SQL statements of "DefaultMsgFilterSQL" and "MsgFilterSQL" are linked logically by "AND" operation if you define additional custom selections by means of "MsgFilterSQL" attribute.

The attribute can be assigned dynamic properties by means of the name DefaultMsgFilterSQL. The data type is STRING.

DefaultPrecision Property

Description

This attribute defines the number of default decimal places, with which the scale value is specified. Write/Read access.

DefaultRulerPrecision Property

Description

This attribute defines the number of decimal places as standard value with which a measured value should be displayed when it is determined using the "Display value at this position" function. Write/Read access.
DefaultSort property

Default sorting order - DefaultSort

Defines the default sorting order in table columns.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ascending</td>
<td>The list is updated starting with the bottom line.</td>
</tr>
<tr>
<td>1</td>
<td>Descending</td>
<td>The list is updated starting with the top line.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name DefaultSort. The data type is LONG.

DefaultSort2 property

DefaultSort2

Use this function to define the sorting method in table columns if not using the default "Date/time/number" sorting order. Instead, you defined a message block in the "DefaultSort2Column" object property to sort the columns based on the "message block/date/time/number" order.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ascending</td>
<td>The list is updated starting with the bottom line.</td>
</tr>
<tr>
<td>1</td>
<td>Descending</td>
<td>The list is updated starting with the top line.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name DefaultSort2. The data type is LONG.

DefaultSort2Column property

DefaultSort2Column

Use this function to define the sorting method in table columns if not using the default "Date/time/number" sorting order.

Define a message block by its object name.

The table columns are now sorted based on the "message block/date/time/number" order.

The attribute can be assigned dynamic properties by means of the name DefaultSort2Column. The data type is STRING.
DeleteData Property

Description
Deletes data in the data buffer of the current trend.
TRUE : All trend data is deleted.
FALSE : The value pair at the "DataIndex" position are deleted.

Note
The property is only supported for the controls prior to WinCC V7.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

Delta Property

Description
Defines or returns the value difference between two main scale graduation marks. Write/Read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

DesiredCurveColor Property

Description
Defines the color of a setpoint trend which belongs to a trend referenced by the "Index" property. The color is defined as an RGB value. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
DesiredCurveCurveForm Property

Description
Defines the form of representation of a setpoint trend which belongs to a trend referenced by the "Index" property. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

0x00000011 Measuring points are connected by a solid line via a step curve
0x00000012 Representation of the measuring points
0x00000014 Measuring points are connected linearly with a solid line
0x00000021 The area under the linearly connected trend is filled.
0x00000022 The area under the stepped curve is filled.
0x00000031: Measuring points are connected by a dashed line via a step curve
0x00000032: Measuring points are connected linearly with a dashed line

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

DesiredCurveSourceNumberOfUAValues Property

Description
Defines the number of value pairs of a setpoint trend which belongs to a trend referenced by the "Index" property. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

DesiredCurveSourceUAArchive Property

Description
Defines the name of the user archive from which the value of a setpoint trend, which belongs to a trend referenced by "Index", is read. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.
See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**DesiredCurveSourceUAArchiveStartID Property**

**Description**

Defines the starting point for the value of a setpoint trend, which belongs to a trend referenced by "Index", from which the values should be read from the archive. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**DesiredCurveSourceUAColumnX Property**

**Description**

Defines the column in the user archive from which the X-values of a setpoint trend, which belongs to a trend referenced by "Index", should be read. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**DesiredCurveSourceUAColumnY Property**

**Description**

Defines the column in the user archive from which the Y-values of a setpoint trend, which belongs to a trend referenced by "Index", should be read. Whether the information is evaluated is dependent on the value of the "DesiredCurveVisible" property.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
DesiredCurveVisible Property

Description
TRUE, a setpoint trend which belongs to a trend referenced by "Index" should be displayed. BOOLEAN write-read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

Direction Property

Description
Defines or returns the bar direction or the position of the slider object. BOOLEAN write-read access. Value range from 0 to 3.
0 = top
1 = bottom
2 = left
3 = right

See also
Slider (Page 212)
Bar (Page 174)
3D Bar (Page 169)
ScreenItem Object (Page 125)

DisplayName property

Display name (DisplayName)
Specifies the user-defined name of the process picture. The attribute is of type "Multilingual String". You can specify names for all languages installed in WinCC.

The logon name is displayed in the following cases:

<table>
<thead>
<tr>
<th>WinCC Explorer</th>
<th>&quot;Details&quot; view in the data window</th>
<th>&quot;Display name&quot; column</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Large Icons&quot; view in the data window</td>
<td>Process picture preview</td>
</tr>
<tr>
<td>WinCC Runtime</td>
<td>&quot;Favorites&quot; system dialog</td>
<td>Process picture preview</td>
</tr>
</tbody>
</table>

The "Display name" attribute can be dynamized with the "DisplayName" name.
DisplayOptions property

Show messages - DisplayOptions

Select the messages to be displayed.

The following selection options are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All messages</td>
</tr>
<tr>
<td>1</td>
<td>Only displayed messages</td>
</tr>
<tr>
<td>2</td>
<td>Only hidden messages</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `DisplayOptions`. The data type is LONG.

DisplayOptions property (before WinCC V7)

Description

Specifies if a button is assigned to a graphic, text, or both.

0 Picture or text: If a picture exists, the button is assigned with the picture, otherwise it is assigned with text.
1 Graphic and text
2 Text only
3 Graphic only

DoubleClickAction property

Action on double-click - DoubleClickAction

Specifies the action to be executed in Runtime by double-clicking on a message line.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
<td>No action.</td>
</tr>
<tr>
<td>1</td>
<td>Loop-in-alarm</td>
<td>Calls the &quot;Loop-in-alarm&quot; function.</td>
</tr>
<tr>
<td>2</td>
<td>Open comments dialog</td>
<td>Calls the &quot;Comments dialog&quot; button function.</td>
</tr>
<tr>
<td>3</td>
<td>Open Infotext dialog</td>
<td>Calls the &quot;Infotext dialog&quot; button function.</td>
</tr>
<tr>
<td>4</td>
<td>Column-dependent</td>
<td>The action is determined by the column in which you double-clicked.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `DoubleClickAction`. The data type is LONG.
1.14.4.6 E

Edit Property

Description
Activates Editing mode for a cell as long as the "Editable" property has been set to TRUE for the corresponding column.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

Editable Property

Description
The "Index" property references a pair of columns. "Editable" defines whether the column pair should be editable. BOOLEAN write-read access.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

EditAtOnce Property

Description
TRUE, if accessing the field with the <TAB> key permits input immediately and without further action. BOOLEAN write-read access.

See also
Text list (Page 196)
I/O Field (Page 184)
ScreenItem Object (Page 125)
Enabled Property

Function

Enables or disables possible operation of an object or issues the corresponding value. TRUE: Enable operation, FALSE: Operation is disabled.

BOOLEAN write-read access.

Example:

The following example disables all objects in the picture "NewPDL1":

```
'VBS71
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    strName = objScreen.ScreenItems.Item(lngIndex).ObjectName    'Read names of objects
    Set objScrItem = objScreen.ScreenItems(strName)
    objScrItem.Enabled=False    'Lock object
Next
```

See also

Screen Object (Page 131)
ScreenItem Object (Page 125)

EnableDelete property

Delete - EnableDelete

Enables deletion of data from the user archive in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>You can delete data from the user archive in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>You cannot delete data from the user archive in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name EnableDelete. The data type is BOOLEAN.
EnableEdit property

Modify - EnableEdit

Enables editing of the data displayed during runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables editing of data during runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables editing of data during runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name EnableEdit. The data type is BOOLEAN.

EnableInsert property

Add - EnableInsert

Enables insertion of data in the user archive in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>You can add data to the user archive in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>You cannot add data to the user archive in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name EnableInsert. The data type is BOOLEAN.

EnablePopupMenu property

EnablePopupMenu

Specifies if the pop-up menu is enabled in the control.

The attribute can be assigned dynamic properties by means of the name EnablePopupMenu. The data type is BOOLEAN.

EndAngle Property

Description

Defines or returns the end of the object. The information is in counterclockwise direction in degrees, beginning at the 12:00 clock position.

See also

Pie segment (Page 152)
Circular arc (Page 151)
EndTime Property

Description

Online Table Control
The "Index" attribute references a pair of columns. "EndTime" defines the end time for displaying this column pair. Whether the information is evaluated is dependent on the TimeRange" and "CommonTime" properties. Write/Read access.

Online Trend Control
The "Index" attribute references a trend. "EndTime" defines the end time for displaying this trend. Whether the information is evaluated is dependent on the "Autorange", "TimeRange" and "CommonX" properties.

Use the "yyyy-mm-dd hh:mm:ss" format when creating a dynamic time range.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

EndValue Property

Description

The "Index" property references a trend. "EndValue" defines the upper limit of the value range to be displayed for the trend. Whether the information is evaluated is dependent on the "Autorange" and "CommonY" properties.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)
EndX Property

Description
Defines the upper limit of the X-axis of a trend referenced with "Index". Whether the information is evaluated is dependent on the "AutorangeX" and "CommonX" properties.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

EndY Property

Description
Defines the upper limit of the Y-axis of a trend referenced with "Index". Whether the information is evaluated is dependent on the "AutorangeY" and "CommonY" properties.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ErrorDescription Property

Function
Error description of the "LastError" property. The error description is provided in English only.

STRING (read only)
The following error messages are defined:

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;&quot;</td>
<td>OK</td>
</tr>
<tr>
<td>&quot;Operation Failed&quot;</td>
<td>Execution error</td>
</tr>
<tr>
<td>&quot;Variable not found&quot;</td>
<td>Tag error</td>
</tr>
<tr>
<td>&quot;Server down&quot;</td>
<td>Server not available.</td>
</tr>
<tr>
<td>&quot;An error occurred for one or several tags&quot;</td>
<td>Multi Tag Error (Error in one or several tags)</td>
</tr>
</tbody>
</table>

In order that ErrorDescription returns a value, a read process must be executed beforehand. If an error occurs during read or write of several tags using the TagSet object, the error is set to "Multi Tag Error". In order to determine at which tag the error occurred and what type of error it was, the ErrorDescription property of each tag must be analyzed.
Example:

The following example displays the error description for "Tag1":

'VBS72
Dim objTag
Set objTag = HMI Runtime.Tags("Tag1")
objTag.Read
MsgBox objTag.ErrorDescription

The following example adds two tags to the TagSet list and outputs the ErrorDescription property as Trace.

'VBS179
Dim group
Set group = HMI Runtime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMI Runtime.Trace "ErrorDescription: " & group.ErrorDescription & vbCrLf

The ErrorDescription property of a tag contained in the list may be accessed as follows:

HMI Runtime.Trace "ErrorDescription: " & group("Motor1").ErrorDescription & vbCrLf

See also

LastError Property (Page 459)
QualityCode Property (Page 546)
TagSet Object (List) (Page 141)
Tag Object (Page 137)

Exponent Property

Description

TRUE, when the display of numbers should be with exponents (e.g."1.00e+000"). BOOLEAN
write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)
ExportDirectoryChangeable property

Directory can be changed - ExportDirectoryChangeable

Enables changing of the directory for data export in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The data export directory can be changed in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The data export directory cannot be changed in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ExportDirectoryChangeable. The data type is BOOLEAN.

ExportDirectoryname property

Directory - ExportDirectoryname

Defines the directory to which the exported Runtime data is written.

You can select or create the directory using the selection button.

The attribute can be assigned dynamic properties by means of the name ExportDirectoryname. The data type is STRING.

ExportFileExtension property

ExportFileExtension

Defines the extension of the export file.

Only the file name extension "csv" is currently supported.

The attribute can be assigned dynamic properties by means of the name ExportFileExtension. The data type is STRING.

ExportFilename property

File name - ExportFilename

Defines the name of the file which is to receive the exported Runtime data.

The attribute can be assigned dynamic properties by means of the name ExportFilename. The data type is STRING.
**ExportFilenameChangeable property**

*File can be renamed - ExportFilenameChangeable*

Enables renaming of the export file in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The export file can be renamed in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The export file cannot be renamed in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ExportFilenameChangeable`. The data type is BOOLEAN.

**ExportFormatGuid property**

**ExportFormatGuid**

Default assignment of the ID number and export provider.

The attribute can be assigned dynamic properties by means of the name `ExportFormatGuid`. The data type is STRING.

**ExportFormatName property**

**Format - ExportFormatName**

Defines the export file format.

Only the "csv" file format is currently available for the export.

The attribute can be assigned dynamic properties by means of the name `ExportFormatName`. The data type is STRING.

**ExportParameters property**

**ExportParameters**

Specifies the parameters of the selected format by means of the properties dialog.

The attribute can be assigned dynamic properties by means of the name `ExportParameters`. The data type is VARIANT.

**ExportSelection property**

**Scope of data export - ExportSelection**

Specifies the control's Runtime data to be exported.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>all</td>
<td>All Runtime data of the control is exported.</td>
</tr>
<tr>
<td>1</td>
<td>Selection</td>
<td>Selected Runtime data of the control is exported.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ExportSelection`. The data type is LONG.

**ExportShowDialog property**

**Show dialog - ExportShowDialog**

Enables the display of the export dialog during runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The dialog is displayed during runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The dialog is not displayed during runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ExportShowDialog`. The data type is BOOLEAN.

**ExportXML property**

**ExportXML**

Only used internally.

The attribute can be assigned dynamic properties by means of the name `ExportXML`.

**ExtendedOperation Property**

**Description**

TRUE, when the slider regulator is set at the respective end value (minimum/maximum value). This is done by clicking the mouse in an area outside the current regulator setting. BOOLEAN write-read access.

**See also**

Slider (Page 212)
ScreenItem Object (Page 125)
ExtendedZoomingEnable Property

Description
Activates/deactivates the ExtendedZooming properties of a picture.
Using ExtendedZooming, the view of a process picture in Runtime may be enlarged or reduced
by using the mouse wheel.
BOOLEAN write-read access.

Example:
Activates ExtendedZooming for picture NewPDL1.

'VBS155
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
objScreen.ExtendedZoomingEnable = 1

See also
Screen Object (Page 131)

1.14.4.7 F
Fe - Fi

FeatureFullscreen property

FeatureFullscreen
Specifies if the "Full screen" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeatureFullscreen.
The data type is BOOLEAN.

FeaturePause property

FeaturePause
Specifies if the "Pause" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeaturePause. The data type is BOOLEAN.
FeaturePlay property

FeaturePlay

Specifies if the "Play" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeaturePlay. The data type is BOOLEAN.

FeatureStepBackward property

FeatureStepBackward

Specifies if the "Step backward" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeatureStepBackward. The data type is BOOLEAN.

FeatureStepForward property

FeatureStepForward

Specifies if the "Step forward" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeatureStepForward. The data type is BOOLEAN.

FeatureStop property

FeatureStop

Specifies if the "Stop" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeatureStop. The data type is BOOLEAN.

FeatureVolume property

FeatureVolume

Specifies if the "Volume" function is available in the control.
The attribute can be assigned dynamic properties by means of the name FeatureVolume. The data type is BOOLEAN.
**FileName property**

**FileName**

Specifies the file whose content you want to display or play.

The attribute can be assigned dynamic properties by means of the name `FileName`. The data type is STRING.

**FillColor Property**

**Description**

Defines or returns the fill pattern color for the object.

LONG (write-read access)

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Enter the appropriate decimal value for each of the three RGB values.

**Example:**

`RGB(200, 150, 100)`

**Example:**

The following example defines the fill color for "ScreenWindow1" to blue:

```vbs
'VBS73
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.FillStyle = 131075
objScreen.FillColor = RGB(0, 0, 255)
```

**See also**

- FillStyle Property (Page 419)
- BackColor Property (Page 322)
- ScreenItem Object (Page 125)
Filling Property

Description
TRUE, when the object can be filled by closed border lines (e.g. representing the fill level of a tank). BOOLEAN write-read access.
The fill level of the object is set by means of the "FillingIndex" property.

See also
ScreenItem Object (Page 125)

FillingDirection properties

Filling direction (FillingDirection)
The "Filling direction" attribute specifies the filling direction for an object enclosed in a frame line.

<table>
<thead>
<tr>
<th>Filling Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom to top</td>
<td>The object is filled from bottom to top.</td>
</tr>
<tr>
<td>Top to bottom</td>
<td>The object is filled from top to bottom.</td>
</tr>
<tr>
<td>Left to right</td>
<td>The object is filled from left to right.</td>
</tr>
<tr>
<td>Right to left</td>
<td>The object is filled from right to left.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name FillingDirection. The data type is LONG.

FillingIndex Property

Description
Defines the %age value (related to the height of the object) to which the object with closed border line is to be filled.
The fill level is represented by the current background color. The unfilled background is transparent.

See also
ScreenItem Object (Page 125)

FillStyle Property

Description
Defines or returns the fill pattern for the object.
### LONG (write-read access)

<table>
<thead>
<tr>
<th>Fill pattern</th>
<th>Value</th>
<th>Fill pattern</th>
<th>Value</th>
<th>Fill pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>65536</td>
<td>1048576</td>
<td>196611</td>
<td>196627</td>
<td></td>
</tr>
<tr>
<td>Massiv</td>
<td>0</td>
<td>1048577</td>
<td>196612</td>
<td>196628</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048578</td>
<td>196613</td>
<td>196629</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048579</td>
<td>196614</td>
<td>196630</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048832</td>
<td>196615</td>
<td>196631</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048833</td>
<td>196616</td>
<td>196632</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048834</td>
<td>196617</td>
<td>196633</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048835</td>
<td>196618</td>
<td>196634</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>131072</td>
<td>196619</td>
<td>196635</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>131073</td>
<td>196620</td>
<td>196636</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>131074</td>
<td>196621</td>
<td>196637</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>131075</td>
<td>196622</td>
<td>196638</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>131076</td>
<td>196623</td>
<td>196639</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>196608</td>
<td>196624</td>
<td>196640</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>196609</td>
<td>196625</td>
<td>196641</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>196610</td>
<td>196626</td>
<td>196642</td>
<td></td>
</tr>
</tbody>
</table>
Example

The following example sets the fill pattern for "ScreenWindow1" to transparent:

```vbs
''VBS190
Dim obj
Set obj = ScreenItems("Rectangle1")
obj.FillStyle = 65536
```

See also

-FillColor Property (Page 418)
-BackColor Property (Page 322)
-Screen Object (Page 131)

FillStyle2 Property

Description

Defines or returns the fill style of the bar.

See also

-Bar (Page 174)
-ScreenItem Object (Page 125)

FillStyleAlignment property

Description

Defines the alignment of the fill pattern for the process picture.

- Normal The fill pattern refers to the process picture. In runtime, no scaling is performed when opening the picture.
- Stretched (window) The fill pattern refers to the window in the Graphics Designer. In runtime, scaling is performed when opening the picture.
FilterSQL property

FilterSQL

Defines an SQL statement for a selection of data in the user archive. The attribute can be assigned dynamic properties by means of the name FilterSQL. The data type is STRING.

FineGrid Property

Description

TRUE, when the value axis is scaled by short tick marks. The distance between two short tick marks can be changed using the "FineGridValue" property. BOOLEAN write-read access.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

FineGridValue Property

Description

Defines the distance between two short tick marks in the scale. Whether the information is evaluated is dependent on the value of the "FineGrid" property.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

FineGridValueX Property

Description

Defines the distance between two short tick marks on the X-axes scaling. Whether the information is evaluated is dependent on the value of the "FineGridX" property.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
FineGridValueY Property

Description
Defines the distance between two short tick marks on the Y-axes scaling. Whether the information is evaluated is dependent on the value of the "FineGridX" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

FineGridX Property

Description
TRUE, when the X-axis graduation is scaled by short tick marks. The distance between two short tick marks can be changed using the "FineGridValueX" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

FineGridY Property

Description
TRUE, when the Y-axis graduation is scaled by short tick marks. The distance between two short tick marks can be changed using the "FineGridValueY" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

FlashBackColor Property

Description
TRUE, when flashing of the background is activated. BOOLEAN write-read access
See also

ScreenItem Object (Page 125)

FlashBorderColor Property

Description

TRUE, when flashing of the object lines is activated. BOOLEAN write-read access.

See also

ScreenItem Object (Page 125)

FlashFlashPicture Property

Description

TRUE, when flashing of the flash picture is activated. BOOLEAN write-read access.

See also

Status display (Page 199)

ScreenItem Object (Page 125)

FlashForeColor Property

Description

TRUE, when flashing of the text is activated. BOOLEAN write-read access.

See also

I/O Field (Page 184)

Static text (Page 165)

Text list (Page 196)

Radio box (Page 207)

Check box (Page 204)

Button (Page 201)

ScreenItem Object (Page 125)
FlashPicReferenced Property

Description

TRUE, when the assigned flash picture should be saved. Otherwise, only the associated object reference is saved. Read only access.

See also

Status display (Page 199)
ScreenItem Object (Page 125)

FlashPicTransColor Property

Description

Defines which color of the bitmap object (.bmp, .dib) assigned to the flash picture should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "FlashPicUseTransColor" property is "True".

See also

ScreenItem Object (Page 125)
Status display (Page 199)

FlashPicture Property

Description

Returns the flash picture. Read-only access.
The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.
In this context, the "FlashPicReferenced" property defines whether the flash picture should be saved together with the object status display or referenced.

See also

Status display (Page 199)
ScreenItem Object (Page 125)
FlashPicUseTransColor Property

Description
TRUE, when the configured color ("FlashPicTransColor" property) of the bitmap objects assigned to the flash picture should be set to "transparent". BOOLEAN write-read access.

See also
Status display (Page 199)
ScreenItem Object (Page 125)

FlashRate Property

Description
Defines or returns the flashing frequency for the object. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note
Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)
FlashRateBackColor Property

Description

Defines or returns the flash frequency for the object background. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note

Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.

See also

ScreenItem Object (Page 125)

FlashRateBorderColor Property

Description

Defines or returns the flash frequency for the lines of the object. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note

Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.

See also

ScreenItem Object (Page 125)
FlashRateFlashPic Property

Description
Defines or returns the flash frequency for the status display. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note
Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).
The information in the table is therefore only for orientation purposes.

See also
Status display (Page 199)
ScreenItem Object (Page 125)

FlashRateForeColor Property

Description
Defines or returns the flash frequency for the object label. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.5 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 2 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 8 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note
Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).
The information in the table is therefore only for orientation purposes.
See also

Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)

Flip property

Flip (Flip)
Specifies flipping of the icon at runtime.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>The icon is not flipped.</td>
</tr>
<tr>
<td>1</td>
<td>Horizontal</td>
<td>The object is flipped along the horizontal center axis.</td>
</tr>
<tr>
<td>2</td>
<td>Vertical</td>
<td>The object is flipped along the vertical center axis.</td>
</tr>
<tr>
<td>3</td>
<td>Both</td>
<td>The object is flipped along the horizontal and vertical center axes.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name Flip. The data type is LONG.

Flip Property

Description

Mirrors the icon on the vertical and/or horizontal middle axis of the icon.

- Zero - 0: The icon is not mirrored.
- Horizontal - 1: The icon is mirrored on the vertical center axis.
- Vertical - 2: The icon is mirrored on the horizontal, center axis.
- Both - 3: The icon is mirrored both on the horizontal and vertical center axes.

See also

HMI Symbol Library (Page 239)
ScreenItem Object (Page 125)
FocusColor Property

Description
If the focus is positioned on the control in Runtime, the labeling and position text are identified by a border. FocusColor defines the color of the border.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

FocusRect Property

Description
TRUE, when the button should be provided with a selection border, in Runtime, as soon as it receives the focus. BOOLEAN write-read access.

See also
WinCC Push Button Control (Page 269)
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

FocusWidth Property

Description
If the focus is positioned on the control in Runtime, the labeling and position text are identified by a border. FocusWidth defines the width of the border, value range of 1-10 pixels. LONG write-read access.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)
Font Property

Name - Font

Sets the font.
The attribute cannot be dynamized.

Font property (before WinCC V7)

Description

Defines or returns the font. Write/Read access.
The font object has the following sub-properties
- Size (Font Size)
- Bold (yes/no)
- Name (font name)
- Italic (yes/no)
- Underline (underline yes/no)
- StrikeThrough (yes/no)

If two font properties are directly assigned, only the default property "Name" is assumed.

Example:

'VBS74
Dim objControl1
Dim objControl2
Set objControl1 = ScreenItems("Control1")
Set objControl2 = ScreenItems("Control2")
objControl2.Font = objControl1.Font ' take over only the type of font

See also

WinCC Slider Control (Page 277)
WinCC Push Button Control (Page 269)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Digital/Analog Clock (Page 246)
FontBold Property

Description
TRUE, when the text in the object should be assigned the "bold" attribute. BOOLEAN write-read access.

See also
WinCC Push Button Control (Page 269)
Group Display (Page 194)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
Bar (Page 174)
ScreenItem Object (Page 125)

FontItalic Property

Description
TRUE, when the text in the object should be assigned the "italic" attribute. BOOLEAN write-read access.

See also
WinCC Push Button Control (Page 269)
Group Display (Page 194)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
Bar (Page 174)
ScreenItem Object (Page 125)
FontName Property

Description
Defines or returns the font name of the text in the object. All the fonts installed in Windows are available for selection.

See also
- WinCC Push Button Control (Page 269)
- Group Display (Page 194)
- Static text (Page 165)
- Text list (Page 196)
- Radio box (Page 207)
- Check box (Page 204)
- Button (Page 201)
- I/O Field (Page 184)
- Bar (Page 174)
- ScreenItem Object (Page 125)

FontPosition Property

Description
Returns the font name for the display of the slider position in the bottom part of the object. All the fonts installed in Windows are available for selection. Read only access.

See also
- WinCC Slider Control (Page 277)
- ScreenItem Object (Page 125)

FontSize Property

Description
Defines or returns the font size of the text in the object in points.

See also
- WinCC Push Button Control (Page 269)
- Group Display (Page 194)
FontStrikeThru Property

Description
TRUE, when the text in the object should be assigned the "strikethrough" attribute. BOOLEAN write-read access.

See also
WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

FontUnderline Property

Description
TRUE, when the text in the object should be assigned the "underline" attribute. BOOLEAN write-read access.

See also
WinCC Push Button Control (Page 269)
Group Display (Page 194)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)
ForeColor Property

Description
Defines or returns the color of the font for the text in the object. LONG write-read access.

See also
WinCC Slider Control (Page 277)
WinCC Push Button Control (Page 269)
WinCC Digital/Analog Clock (Page 246)
HMI Symbol Library (Page 239)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)

ForeColor property

Foreground color (ForeColor)
Specifies the foreground color of the icon in the "Color selection" dialog. The icon is displayed in the foreground color if the "Shadow" and "Solid" foreground mode is set.
The attribute can be assigned dynamic properties by means of the name ForeColor. The data type is LONG.

ForeFlashColorOff Property

Description
Defines or returns the color of the text for flash status "Off". LONG write-read access.

See also
Text list (Page 196)
Static text (Page 165)
Radio box (Page 207)
Check box (Page 204)
ForeFlashColorOn Property

Description
Defines or returns the color of the text for flash status "On". LONG write-read access.

See also
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)

FrameColor Property

Description
Defines or returns the color of the rectangular or square area located on the graduated scale disk. LONG write-read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

FrameColorDown Property

Description
Defines or returns the color for the right, bottom part of the 3D frame of the button (button pressed). LONG write-read access.
See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

FrameColorUp Property

Description
Defines or returns the color for the left, top part of the 3D frame of the button (button not pressed). LONG write-read access.

See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

FramePicture Property

Description
Returns the picture name of the background picture for the graduated scale disk. Read only access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

FrameScale Property

Description
Defines or returns the diameter of the graduated scale disk in relation to smallest value of the width and height geometric attributes. Write/Read access.
The value range is (scale distance - scale width) to 1.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)
FrameWidth Property

Description

Defines or returns the border width of the button in pixels. Write/Read access.

See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

FreezeProviderConnections Property

Description

Enables modification of the data connection properties ("ProviderType", "Source"...), without
the change being effective immediately. On changing "SourceTagNameX", for example,
impermissible combinations can created with "SourceTagNameY".

Therefore, FreezeProviderConnections" must be set to TRUE before modifying a data
connection attribute. After modifying all the data connection, "FreezeProviderConnection" is
set to FALSE and the changes take effect.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

1.14.4.8 G

GlobalColorScheme property

Description

Defines whether the colors defined for the current design in the global color scheme will be
used for this object.

TRUE if the object is displayed with the colors from the global color scheme defined for this
object type.

FALSE if the object is displayed with the colors as per the settings in the object.

BOOLEAN write-read access.
GlobalShadow property

Description
Defines whether the object will be displayed with the shadowing defined in the active design. TRUE if the object is displayed with the global shadow defined for this object type. FALSE if no shadow is displayed. BOOLEAN write-read access.

GraphDirection property (before WinCC V7)

Description
Defines which edge of the trend window should display the current values. Write/Read access.
0: Positive values run to the right and upwards.
-1: Positive values run to the left and upwards.
-2: Positive values run to the right and upwards.
-3: Positive values run to the right and downwards.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

GraphDirection Property

Write direction - GraphDirection
Defines the direction of the update of axis values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>From the right</td>
<td>The updated values are displayed starting at the right in the control.</td>
</tr>
<tr>
<td>1</td>
<td>From the left</td>
<td>The updated values are displayed starting at the left in the control.</td>
</tr>
<tr>
<td>2</td>
<td>From the top</td>
<td>The updated values are displayed starting at the top in the control.</td>
</tr>
<tr>
<td>3</td>
<td>From the bottom</td>
<td>The updated values are displayed starting at the bottom in the control.</td>
</tr>
</tbody>
</table>

True type fonts must be used within the trend window if "From the top" or "From the bottom" is selected for write direction. Only this setting ensures legibility of the labeling of the vertical axis.

The attribute can be assigned dynamic properties by means of the name GraphDirection. The data type is LONG.
GridLineColor property

Color of the row divider / content - GridLineColor

Defines the color of row/column dividers in table contents. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name GridLineColor. The data type is LONG.

GridLineHorz Property

Description

TRUE, when the message window columns are separated by horizontal dividing lines. BOOLEAN write-read access.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

GridLines Property

Description

TRUE, when the trend window is displayed with grid lines parallel to the X-axis. The distance between two grid lines can be changed using the "GridLineValue" property. BOOLEAN write-read access.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

GridlinesValueX Property

Description

Defines or returns the distance between two grid lines on the X-axis. Whether the information is evaluated is dependent on the value of the "GridLinesX" property.
See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**GridlinesValueY Property**

**Description**
Defines or returns the distance between two grid lines on the Y-axis. Whether the information is evaluated is dependent on the value of the "GridLinesY" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**GridlinesX Property**

**Description**
TRUE, when the trend window is displayed with grid lines parallel to the X-axis. The distance between two grid lines can be changed using the "GridLineValueX" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**GridlinesY Property**

**Description**
TRUE, when the trend window is displayed with grid lines parallel to the Y-axis. The distance between two grid lines can be changed using the "GridLineValueX" property.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
GridLineValue Property

Description
Defines the distance between two grid lines. Whether the information is evaluated is dependent on the value of the "GridLines" property.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

GridLineVert Property

Description
TRUE, when the message window columns are separated by vertical dividing lines. BOOLEAN write-read access.

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

GridLineWidth property

Width of dividers - GridLineWidth
Defines the line weight of the row/column dividers in pixels.

The attribute can be assigned dynamic properties by means of the name GridLineWidth. The data type is LONG.

1.14.4.9 H
Ha - Hi

HandFillColor Property

Description
Defines or returns the fill color of all the hands in the analog clock. In order that the hands are displayed with the fill color defined, the "Handtype" property must be set to "0" (covering). LONG write-read access.
See also

WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

Handtype Property

Description
Defines the representation of the hands:

- 0: The hands are filled in the hand color defined and the edges in the foreground color.
- 1: The hands fill color is transparent and the edges displayed in the foreground color.

See also

WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

HeaderSort Property

Description
Specifies if sorting of messages by message block column header is possible.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Height Property

Description
Defines or returns the height of the object in pixels.

LONG (write-read access)
Example:

The following example halves the height of all objects in the "NewPDL1" picture whose names begin with "Circle":

```vbs
'VBS75
Dim objScreen
Dim objCircle
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    'Searching all circles
    strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
    If "Circle" = Left(strName, 6) Then
        'to halve the height of the circles
        Set objCircle = objScreen.ScreenItems(strName)
        objCircle.Height = objCircle.Height / 2
    End If
Next
```

See also

- Width Property (Page 698)
- Object types of the ScreenItem object (Page 143)
- ScreenItem Object (Page 125)

HiddenInput Property

Description

TRUE, when the input value should not be displayed when being entered. Each character entered is substituted by a *. BOOLEAN write-read access.

See also

- I/O Field (Page 184)
- ScreenItem Object (Page 125)
**HideTagNames Property**

**Description**

TRUE if the archive and tag name in the trend should be hidden via the right mouse button, in the status line and in the table to display the coordinates. BOOLEAN write-read access.

**HitlistColumnAdd property**

**HitlistColumnAdd**

Transfers the selected message block from the list of available message blocks to the list of selected message blocks.

The attribute can be assigned dynamic properties by means of the name `HitlistColumnAdd`. The data type is STRING.

**HitlistColumnAutoAdjust property**

**HitlistColumnAutoAdjust**

Automatically optimizes the width of the columns in the hit list in Runtime. If the configured column is not sufficient for the displayed text, the column width is increased.

If the user changes the width of columns in Runtime, the width is only optimized again when the picture is opened again.

The attribute can be assigned dynamic properties using the name `HitlistColumnAutoAdjust`. The data type is BOOLEAN.

**See also**

WinCC AlarmControl (Page 241)

HitlistColumn object (list) (Page 222)

**HitlistColumnCount property**

**HitlistColumnCount**

Specifies the number of message blocks displayed in the hitlist in Runtime.

The attribute can be assigned dynamic properties by means of the name `HitlistColumnCount`. The data type is LONG.
HitlistColumnIndex property

**HitlistColumnIndex**

References a message block selected for the hitlist. Using this attribute you can assign the values of other attributes to a specific message block of the hitlist.

Values between 0 and "HitlistColumnCount" minus 1 are valid for "HitlistColumnIndex". Attribute "HitlistColumnCount" defines the number of message blocks selected for the hitlist.

The "HitlistColumnIndex" attribute can be assigned dynamic properties by means of attribute **HitlistColumnRepos**. The data type is LONG.

HitlistColumnName property

**HitlistColumnName**

Displays the name of the message block of the hitlist which is referenced with attribute "HitlistColumnIndex". You cannot edit this name.

The attribute can be assigned dynamic properties by means of the name **HitlistColumnName**. The data type is STRING.

HitlistColumnRemove property

**HitlistColumnRemove**

Cuts the marked message block from the list of selected message blocks and pastes it to the list of available message blocks.

The attribute can be assigned dynamic properties by means of the name **HitlistColumnRemove**. The data type is STRING.

HitlistColumnRepos

**Up/Down - MessageColumnRepos/HitlistColumnRepos**

Resorts the message blocks. The "Up" and "Down" commands move the selected message block accordingly in the list. This moves the message block in Runtime Control towards the front or towards the back.

The attribute for the hitlist can be assigned dynamic properties by means of the name **HitlistColumnRepos**.

The attribute for the message list can be assigned dynamic properties by means of the name **MessageColumnRepos**.

The data type is LONG.
HitlistColumnSort property

HitlistColumnSort

Defines the sorting order of the message block referenced in "HitlistColumnIndex" for the hitlist.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
<td>No sorting</td>
</tr>
<tr>
<td>1</td>
<td>Ascending</td>
<td>Ascending order, starting at the lowest value.</td>
</tr>
<tr>
<td>2</td>
<td>Descending</td>
<td>Descending order, starting at the highest value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HitlistColumnSort`. The data type is LONG.

HitlistColumnSortIndex property

HitlistColumnSortIndex

Defines the sorting order of the message block referenced in "HitlistColumnIndex" in the hitlist.

The sorting criterion is removed from "HitlistColumnSort" if you set a "0" value.

The attribute can be assigned dynamic properties by means of the name `HitlistColumnSortIndex`. The data type is LONG.

HitlistColumnVisible

Selected message blocks - MessageColumnVisible/HitlistColumnVisible

Selected message blocks of message list or hitlist that are displayed in Runtime. Defines whether the message block referenced in "MessageColumnIndex" or "HitlistColumnIndex" is displayed.

The attribute for the message list can be assigned dynamic properties by means of the name `MessageColumnVisible`.

The attribute for the hitlist can be assigned dynamic properties by means of the name `HitlistColumnVisible`.

The data type is BOOLEAN.

HitlistDefaultSort property

HitlistDefaultSort

Defines the default sorting order in the table columns of the hitlist.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ascending</td>
<td>The list is sorted in ascending order based on frequency.</td>
</tr>
<tr>
<td>1</td>
<td>Descending</td>
<td>The list is sorted in descending order based on frequency.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HitlistDefaultSort`. The data type is LONG.

**HitListMaxSourceItems property**

**Maximum number of data records - HitListMaxSourceItems**

Defines the maximum number of data records for statistics.

The attribute can be assigned dynamic properties by means of the name `HitListMaxSourceItems` . The data type is LONG.

**HitListMaxSourceItemsWarn property**

**Warning when maximum is reached - HitListMaxSourceItemsWarn**

Enables the output of a warning notice after the valid number of data records was reached.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>A warning is output after the valid maximum number of data records was reached.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A warning is not output after the valid maximum number of data records was reached.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HitListMaxSourceItemsWarn`. The data type is BOOLEAN.

**HitListRelTime property**

**Time range for statistics - HitListRelTime**

Sets a time range for the statistics.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The time range set for statistics is used if this range was not defined in the selection.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time range is not used.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HitListRelTime`. The data type is BOOLEAN.
HitListRelTimeFactor property

**Time range - HitListRelTimeFactor**

Defines the factor for calculating the time range. Only integer factors are valid.

The attribute can be assigned dynamic properties by means of the name `HitListRelTimeFactor`. The data type is LONG.

HitListRelTimeFactorType property

**Time range - HitListRelTimeFactorType**

Defines the time unit for calculating the time range.

The following time units are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Minute</td>
</tr>
<tr>
<td>1</td>
<td>Hour</td>
</tr>
<tr>
<td>2</td>
<td>Day</td>
</tr>
<tr>
<td>3</td>
<td>Week</td>
</tr>
<tr>
<td>4</td>
<td>Month</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HitListMaxRelTimeFactorType`. The data type is LONG.

Ho - Hy

HorizontalGridLines property

**Horizontal - HorizontalGridLines**

Defines whether horizontal separating lines will be displayed.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of horizontal dividers.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of horizontal dividers.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `HorizontalGridLines`. The data type is BOOLEAN.
Hotkey Property

Description

Returns the function key related to the mouse operation in respect of a button object. Read only access.

See also

Button (Page 201)
ScreenItem Object (Page 125)

HourNeedleHeight Property

Description

Defines or returns the length of the hour hand for the analog clock. The specification of the length is entered as a percentage value in relation to half the length of the short side of the rectangular background. Write/Read access.

Example:
The shorter side of the rectangular background is 100 pixels long. The hour hand length is 50. This results in a length of the hour hand of \( (100 \text{ pixels} / 2) \times 0.5 = 25 \text{ pixels} \).

See also

WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

HourNeedleWidth Property

Description

Defines or returns the width of the hour hand for the analog clock. The width is specified as a percentage value related to double the length of the hour hand. Write/Read access.

Example:
The length of the hour hand is 25 pixels. The hour hand width is 10. This results in a width of the hour hand of \( 25 \text{ pixels} \times 2 \times 0.1 = 5 \text{ pixels} \).

See also

WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)
Hysteresis Property

Description

TRUE, when the display should appear with hysteresis. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

HysteresisRange Property

Description

Defines the hysteresis in % of the displayed value or returns it.
The Hysteresis property must be set to TRUE for the hysteresis to be calculated.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

1.14.4.10

IconSpace property

IconSpace

Defines the spacing between the icons and text in the table cells. The value is active if and icon and text are displayed.
The attribute can be assigned dynamic properties by means of the name IconSpace. The data type is LONG.

IndependentWindow property

Description

Defines whether the display of the picture window in Runtime depends on the process picture in which the picture window was configured.
TRUE if the size and position of the picture window are independent of the process picture and only defined by the "Window mode" attribute.
FALSE if the size and position of the picture window change with the shift or scaling of the process picture.

**Index Property**

**Description**

**Check box, radio box**
Defines or returns the number (0 to 31) of the field whose text is to be defined.

**Combo box, list box**
Defines or returns the number (0 to 31) of the line whose text is to be defined.

**Polygon, polyline, tube polygon**
Defines or returns the number of the corner point whose position coordinates are to be modified or displayed.

**WinCC online trend control, WinCC online table control, WinCC function trend control**
The "Index" property is evaluated by other properties in order to be able to assign the settings to a specific trend or column pair. The valid values for the index move within the range from 0 to (NumItems - 1). The "NumItems" properties contains the number of the trends/column pairs to be displayed. The index must always be set before you change the properties of a trend / column in runtime.

**Status display**
Defines the status (0 to 255) or returns it. A basic picture and flash picture can be defined for each status value.

**See also**

Status display (Page 199)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
Polyline (Page 158)
Polygon (Page 156)
Radio box (Page 207)
Check box (Page 204)
ScreenItem Object (Page 125)
**InnerBevelOffset Property**

**Description**
Defines the distance between the inner and outer bevels.

**See also**
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

**InnerBevelStyle Property**

**Description**
Defines the 3D effect for the inner bevel of the object.

- 0: No border.
- 1: The border is displayed depressed.
- 2: The border is displayed raised.
- 3: The border is displayed in one color without a 3D effect. The border color is defined by the "BevelColorDown" property.

**See also**
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

**InnerBevelWidth Property**

**Description**
Defines the width of the inner bevel in pixels.

**See also**
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)
InputValue property

Description

Defines the value to be entered by the user in the I/O field. The value is not displayed in the I/O field when the property is set.

If you want the value to be displayed in the I/O field after confirmation with the <Return> key, configure a direct connection between the properties "input value" and "output value". The direct connection is only practical when no tag is connected to the output value, but the user can nevertheless query the specified value, for example, through a script.

LONG write-read access.

See also

Example: Calling Methods of an ActiveX Control (Page 841)

InsertData Property

Description

Inserts data for the current trend.

TRUE : "DataIndex" is ignored and the data is appended to that in the data buffer.
FALSE : The data is inserted at the "DataIndex" position in the data buffer.

The trend window is redrawn following each operation involving "Insert Data".

Note

The property is only supported for the controls prior to WinCC V7.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

Instance property

Description

Returns an instance of the alarm object.

See also

Alarms object (list) (Page 110)
ItemBorderBackColor Property

Description
Defines or returns the background color for dividing lines in the selection list of the text list object. LONG write-read access. The background color is only visible with the property setting ItemBorderStyle > 0.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

ItemBorderColor Property

Description
Defines or returns the color for dividing lines in the selection list of the text list object. LONG write-read access.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

ItemBorderStyle Property

Description
Defines or returns the color for the dividing line style in the selection list of the text list object. Value range from 0 to 4.

See also
Text list (Page 196)
ScreenItem Object (Page 125)
**ItemBorderWidth Property**

**Description**

Defines or returns the dividing line weight in pixels in the selection list of the text list object.

**See also**

Text list (Page 196)
ScreenItem Object (Page 125)

**ItemProviderClsid Property**

**Description**

"ItemProviderClsid" shows, if the trend referenced using Index in Trend Control is connected with an archive tag or an online tag.

Notice: If you assign a value to the "ProviderClsid" property, you will overwrite the trend-specific property "ItemProviderClsid".

- {416A09D2-8B5A-11D2-8B81-006097A45D48}: The trend is connected to an archive tag.
- {A3F69593-8AB0-11D2-A440-00A0C9DBB64E}: The trend is connected to an online tag.

If the trends are being supplied with archive and online tags, the property "ProviderClsid" returns the value "{00000000-0000-0000-0000-000000000000}".

**ItemVisible Property**

**Description**

TRUE, when a trend or a column pair reference by the "Index" property is visible. BOOLEAN write-read access.

**See also**

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
Lab - Las

Label Property

Description
The "Index" property references a trend. Label is used to define the name of the time axis or value axis in accordance with the value of the "TimeAxis" property.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

LabelColor Property

Description
Defines the color of the scale label.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

LabelX Property

Description
Defines or returns the label on the X-axis for a trend referenced by "Index" according to the value of "TimeAxisX". Write/Read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
LabelY Property

Description
Defines or returns the label on the Y-axis for a trend referenced by "Index" according to the value of "TimeAxisY". Write/Read access.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

LanguageSwitch Property

Description
Returns the value which defines where the language dependent assigned texts are stored.
Read only access.

TRUE, when the texts in the Text Library are managed. Translation to other language occurs in the Text Library.
FALSE, when the texts are managed directly in the object. Translation to other language can be carried out using Text Distributor.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

Language Property

Description
Defines the current Runtime language or reads it.

You specify the Runtime language in VBS by using a country code, e.g., 1031 for German - Default, 1033 for English - USA etc. A summary of all country codes may be found in the Basics of VBScript under the subject header "Regional Scheme ID (LCID) Diagram".

INTEGER (write-read access)
Example:

The following example sets the data language to German:

'VBS76
HMIRuntime.Language = 1031

See also

HMIRuntime Object (Page 118)

LastError Property

Description

Returns an error code regarding the success of the last operation, e.g. information on a tag write or read process. The "QualityCode" property can provide information on the quality of the returned value. A description of the error can be called in using the "ErrorDescription" property.

LONG (read only)

The following error codes are defined:

<table>
<thead>
<tr>
<th>Code in hexadecimal notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>OK</td>
</tr>
<tr>
<td>0x80040001</td>
<td>Execution error</td>
</tr>
<tr>
<td>0x80040002</td>
<td>Tag error</td>
</tr>
<tr>
<td>0x80040003</td>
<td>Server not available.</td>
</tr>
<tr>
<td>0x80040004</td>
<td>Multi Tag Error (Error in one or several tags)</td>
</tr>
</tbody>
</table>

In order that LastError returns a value, a read must be executed beforehand.

If an error occurs during read or write of several tags using the TagSet object, the error is set to "Multi Tag Error". In order to determine at which tag the error occurred and what type of error it was, the LastError property of each tag must be analyzed.

Example:

The following example displays the error code for "Tag1":

'VBS77
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.LastError
The following example adds two tags to the TagSet list and outputs the LastError property as Trace.

```vbs
'VBS178
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "LastError: " & group.LastError & vbCrLf

The LastError property of a tag contained in the list may be accessed as follows:

HMIRuntime.Trace "LastError: " & group("Motor1").LastError & vbCrLf
```

See also

TagSet Object (List) (Page 141)
QualityCode Property (Page 546)
ErrorDescription Property (Page 411)
Tag Object (Page 137)

Layer

Layer Property

Description

Returns the layer of the picture in which the object is located. There is a total of 32 layers available, whereby Layer "0" is the bottom layer and Layer "31" the top layer.

The configured objects are initially in the background of a layer.

LONG (read only)

Note

The layer property specifies the layer in which the object is located. The layer "0" is output as "Layer0".

When accessed, the layers are counted up from 1 in VBS. Therefore, the layer "1" must be addressed with "layers(2)".
Example:

The following example displays the name and layer of all the objects in the picture "NewPDL1":

    ' VBS78
    Dim objScreen
    Dim objScrItem
    Dim lngAnswer
    Dim lngIndex
    Dim strName
    lngIndex = 1
    Set objScreen = HMIRuntime.Screens("NewPDL1")
    For lngIndex = 1 To objScreen.ScreenItems.Count
        strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
        Set objScrItem = objScreen.ScreenItems(strName)
        lngAnswer = MsgBox(strName & " is in layer ", conn objScrItem.Layer, vbOKCancel)
        If vbCancel = lngAnswer Then Exit For
    Next

See also

ScreenItem Object (Page 125)

Layer00Checked Property

Description

TRUE, when limit 0 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer00Value and Layer00Color properties.

See also

3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer01Checked Property

Description

TRUE, when limit 1 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer01Value and Layer01Color properties.
Layer02Checked Property

Description
TRUE, when limit 2 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer02Value and Layer02Color properties.

Layer03Checked Property

Description
TRUE, when limit 3 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer03Value and Layer03Color properties.

Layer04Checked Property

Description
TRUE, when limit 4 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer04Value and Layer04Color properties.
Layer05Checked Property

Description
TRUE, when limit 5 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer05Value and Layer05Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer06Checked Property

Description
TRUE, when limit 6 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer06Value and Layer06Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer07Checked Property

Description
TRUE, when limit 7 should be monitored. BOOLEAN write/read access. Limit value and representation are defined with the Layer07Value and Layer07Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer08Checked Property

Description
TRUE, when limit 8 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer08Value and Layer08Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer09Checked Property

Description
TRUE, when limit 9 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer09Value and Layer09Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer10Checked Property

Description
TRUE, when limit 10 should be monitored. BOOLEAN write/read access.
Limit value and representation are defined with the Layer10Value and Layer10Color properties.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer00Color Property

Description
Defines or returns the color for limit 0. LONG write/read access. When monitoring of the limit value is activated (Layer00Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer01Color Property

Description
Defines or returns the color for limit 1. LONG write/read access. When monitoring of the limit value is activated (Layer01Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer02Color Property

Description
Defines or returns the color for limit 2. LONG write/read access. When monitoring of the limit value is activated (Layer02Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer03Color Property

Description
Defines or returns the color for limit 3. LONG write/read access.
When monitoring of the limit value is activated (Layer03Checked property), the bar turns to
the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer04Color Property

Description
Defines or returns the color for limit 4. LONG write/read access.
When monitoring of the limit value is activated (Layer04Checked property), the bar turns to
the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer05Color Property

Description
Defines or returns the color for limit 5. LONG write/read access.
When monitoring of the limit value is activated (Layer05Checked property), the bar turns to
the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer06Color Property

Description
Defines or returns the color for limit 6. LONG write/read access. When monitoring of the limit value is activated (Layer06Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer07Color Property

Description
Defines or returns the color for limit 7. LONG write/read access. When monitoring of the limit value is activated (Layer07Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer08Color Property

Description
Defines or returns the color for limit 8. LONG write/read access. When monitoring of the limit value is activated (Layer08Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer09Color Property

Description
Defines or returns the color for limit 9. LONG write/read access. When monitoring of the limit value is activated (Layer09Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer10Color Property

Description
Defines or returns the color for limit 10. LONG write/read access. When monitoring of the limit value is activated (Layer10Checked property), the bar turns to the color defined by this attribute on reaching the limit value.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer00FillColor property
Bar fill color 0 (Layer00FillColor)
The "Layer00FillColor" attribute defines the color with which the bar is filled in relation to "Limit 0".
The "Layer00FillColor" attribute can be made dynamic with the name "Layer00FillColor".

Layer01FillColor property
Layer01FillColor
The "Layer01FillColor" attribute defines the color with which the bar is filled in relation to "Limit 1".
The "Layer01FillColor" attribute can be made dynamic with the name "Layer01FillColor".

Layer02FillColor property
Layer02FillColor
The "Layer02FillColor" attribute defines the color with which the bar is filled in relation to "Limit 2".

The "Layer02FillColor" attribute can be made dynamic with the name "Layer02FillColor".

**Layer03FillColor property**

Layer03FillColor
The "Layer03FillColor" attribute defines the color with which the bar is filled in relation to "Limit 3".

The "Layer03FillColor" attribute can be made dynamic with the name "Layer03FillColor".

**Layer04FillColor property**

Layer04FillColor
The "Layer04FillColor" attribute defines the color with which the bar is filled in relation to "Limit 4".

The "Layer04FillColor" attribute can be made dynamic with the name "Layer04FillColor".

**Layer05FillColor property**

Layer05FillColor
The "Layer05FillColor" attribute defines the color with which the bar is filled in relation to "Limit 5".

The "Layer05FillColor" attribute can be made dynamic with the name "Layer05FillColor".

**Layer06FillColor property**

Layer06FillColor
The "Layer06FillColor" attribute defines the color with which the bar is filled in relation to "Limit 6".

The "Layer06FillColor" attribute can be made dynamic with the name "Layer06FillColor".

**Layer07FillColor property**

Layer07FillColor
The "Layer07FillColor" attribute defines the color with which the bar is filled in relation to "Limit 7".

The "Layer07FillColor" attribute can be made dynamic with the name "Layer07FillColor".

**Layer08FillColor property**

Layer08FillColor
The "Layer08FillColor" attribute defines the color with which the bar is filled in relation to "Limit 8".
The "Layer08FillColor" attribute can be made dynamic with the name "Layer08FillColor".

Layer09FillColor property

Layer09FillColor
The "Layer09FillColor" attribute defines the color with which the bar is filled in relation to "Limit 9".
The "Layer09FillColor" attribute can be made dynamic with the name "Layer09FillColor".

Layer10FillColor property

Layer10FillColor
The "Layer10FillColor" attribute defines the color with which the bar is filled in relation to "Limit 10".
The "Layer10FillColor" attribute can be made dynamic with the name "Layer10FillColor".

Layer00FillStyle property

Layer00FillStyle
The "Layer00FillStyle" attribute defines the style of the bar in relation to "Limit 0". For the fill pattern to become visible, "bar fill color 0" must differ from "bar color 0". There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed. The "Layer00FillStyle" attribute can be made dynamic with the name "Layer00FillStyle".

Layer01FillStyle property

Layer01FillStyle
The "Layer01FillStyle" attribute defines the style of the bar in relation to "Limit 1". For the fill pattern to become visible, "bar fill color 1" must differ from "bar color 1". There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed. The "Layer01FillStyle" attribute can be made dynamic with the name "Layer01FillStyle".

Layer02FillStyle property

Layer02FillStyle
The "Layer02FillStyle" attribute defines the style of the bar in relation to "Limit 2". For the fill pattern to become visible, "bar fill color 2" must differ from "bar color 2". There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.
The "Layer02FillStyle" attribute can be made dynamic with the name "Layer02FillStyle".

Layer03FillStyle property
Layer03FillStyle
The "Layer03FillStyle" attribute defines the style of the bar in relation to "Limit 3". For the fill pattern to become visible, "bar fill color 3" must differ from "bar color 3".
There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.
The "Layer03FillStyle" attribute can be made dynamic with the name "Layer03FillStyle".

Layer04FillStyle property
Layer04FillStyle
The "Layer04FillStyle" attribute defines the style of the bar in relation to "Limit 4". For the fill pattern to become visible, "bar fill color 4" must differ from "bar color 4".
There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.
The "Layer04FillStyle" attribute can be made dynamic with the name "Layer04FillStyle".

Layer05FillStyle property
Layer05FillStyle
The "Layer05FillStyle" attribute defines the style of the bar in relation to "Limit 5". For the fill pattern to become visible, "bar fill color 5" must differ from "bar color 5".
There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.
The "Layer05FillStyle" attribute can be made dynamic with the name "Layer05FillStyle".

Layer06FillStyle property
Layer06FillStyle
The "Layer06FillStyle" attribute defines the style of the bar in relation to "Limit 6". For the fill pattern to become visible, "bar fill color 6" must differ from "bar color 6".
There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.
The "Layer06FillStyle" attribute can be made dynamic with the name "Layer06FillStyle".

Layer07FillStyle property
Layer07FillStyle
The "Layer07FillStyle" attribute defines the style of the bar in relation to "Limit 7". For the fill pattern to become visible, "bar fill color 7" must differ from "bar color 7".
There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.

The "Layer07FillStyle" attribute can be made dynamic with the name "Layer07FillStyle".

**Layer08FillStyle property**

Layer08FillStyle

The "Layer08FillStyle" attribute defines the style of the bar in relation to "Limit 8". For the fill pattern to become visible, "bar fill color 8" must differ from "bar color 8".

There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.

The "Layer08FillStyle" attribute can be made dynamic with the name "Layer08FillStyle".

**Layer09FillStyle property**

Layer09FillStyle

The "Layer09FillStyle" attribute defines the style of the bar in relation to "Limit 9". For the fill pattern to become visible, "bar fill color 9" must differ from "bar color 9".

There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.

The "Layer09FillStyle" attribute can be made dynamic with the name "Layer09FillStyle".

**Layer10FillStyle property**

Layer10FillStyle

The "Layer10FillStyle" attribute defines the style of the bar in relation to "Limit 10". For the fill pattern to become visible, "bar fill color 10" must differ from "bar color 10".

There is a choice of 50 fill styles. The 0 "Solid" fill style fills the object with the set background color. The 1 "Transparent" fill style means neither a background nor a fill pattern is displayed.

The "Layer10FillStyle" attribute can be made dynamic with the name "Layer10FillStyle".

**Layer00Value Property**

**Description**

Determines the value for "Limit 0" or returns it.

Monitoring only takes effect when the Layer00Checked property value is set to TRUE.

**See also**

3D Bar (Page 169)
ScreenItem Object (Page 125)
Layer01Value Property

Description
D Determines the value for "Limit 1" or returns it. Monitoring only takes effect when the Layer01Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer02Value Property

Description
D Determines the value for "Limit 2" or returns it. Monitoring only takes effect when the Layer02Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer03Value Property

Description
D Determines the value for "Limit 3" or returns it. Monitoring only takes effect when the Layer03Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer04Value Property

Description
D Determines the value for "Limit 4" or returns it. Monitoring only takes effect when the Layer04Checked property value is set to TRUE.
See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer05Value Property

Description
Determines the value for "Limit 5" or returns it.
Monitoring only takes effect when the Layer05Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer06Value Property

Description
Determines the value for "Limit 6" or returns it.
Monitoring only takes effect when the Layer06Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer07Value Property

Description
Determines the value for "Limit 7" or returns it.
Monitoring only takes effect when the Layer07Checked property value is set to TRUE.

See also
ScreenItem Object (Page 125)
3D Bar (Page 169)
Layer08Value Property

Description
Determines the value for "Limit 8" or returns it.
Monitoring only takes effect when the Layer08Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer09Value Property

Description
Determines the value for "Limit 9" or returns it.
Monitoring only takes effect when the Layer09Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

Layer10Value Property

Description
Determines the value for "Limit 10" or returns it.
Monitoring only takes effect when the Layer10Checked property value is set to TRUE.

See also
3D Bar (Page 169)
ScreenItem Object (Page 125)

LayerDeclutteringEnable Property

Description
Returns the LayerDecluttering properties of a picture.
LayerDecluttering enables fading in and out of layers depending on the set minimum and maximum zoom.
BOOLEAN Read-only access.

Example:
The example outputs the LayerDecluttering Property NewPDL1 as a trace.

'VBS156
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine

See also
Screen Object (Page 131)

Layers Property

Description
Returns an object of type "Layers".
Layers (read-only)

See also
Layers Object (Listing) (Page 121)
HMIRuntime Object (Page 118)

Le - Li

Left Property

Description
Defines or returns the X-coordinate of an object (measured from the top left edge of the picture) in pixels. The X-coordinate relates to the top left corner of the rectangle enclosing the object.
LONG (write-read access)
Example:

The following example shifts all objects in the picture "NewPDL1" 5 pixels to the left:

```vbs
'VBS79
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
objScrItem.Left = objScrItem.Left - 5
Next
```

See also

Top Property (Page 643)
ScreenItem Object (Page 125)

LeftComma Property

Description

Defines or returns the number of digits to the left of the decimal point (0 to 20).

See also

Bar (Page 174)
ScreenItem Object (Page 125)

LightEffect Property

Description

TRUE, when the light effect should be activated. BOOLEAN write-read access.

See also

3D Bar (Page 169)
ScreenItem Object (Page 125)
LimitHigh4 Property

Description
Determines the upper limit value for "Reserve 4" or returns it. The CheckLimitHigh4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored. The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh4 property.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

LimitHigh5 Property

Description
Determines the upper limit value for "Reserve 5" or returns it. The CheckLimitHigh5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored. The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh5 property.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

LimitLow4 Property

Description
Determines the lower limit value for "Reserve 4" or returns it. The CheckLimitLow4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored. The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow4 property.

See also
Bar (Page 174)
ScreenItem Object (Page 125)
LimitLow5 Property

Description
Determines the lower limit value for "Reserve 5" or returns it.
The CheckLimitLow5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow5 property.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

LimitMax Property

Description
Determines the upper limit value as an absolute value depending on the data format or returns it.
If the displayed value exceeds the upper limit value, it is displayed by a sequence of *** (not displayable).

See also
I/O Field (Page 184)
ScreenItem Object (Page 125)

LimitMin Property

Description
Determines the lower limit value as an absolute value depending on the data format or returns it.
If the displayed value exceeds the upper limit value, it is displayed by a sequence of *** (not displayable).

See also
I/O Field (Page 184)
ScreenItem Object (Page 125)
LineColor property

Color of window dividers - LineColor

Specifies the color of the window dividers. The button opens the "Color selection" dialog. The attribute can be assigned dynamic properties by means of the name LineColor. The data type is LONG.

LineFont Property

Description

TRUE, when the font size should be automatically adapted to the line height. BOOLEAN write-read access.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

LineHeight Property

Description

TRUE, when the line height can be modified. BOOLEAN write-read access.

The "LineHeight" property is only deactivated if both properties "LineHeight" and "LineFont" are set to "FALSE".

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

LineJoinStyle property

Description

Defines the way that corners are displayed in a tube polygon.

Angle The tubes are joined at corner points without rounding
Round The tubes are rounded at the outside corner points.
LineTitle Property

Description
TRUE, when the message window a column with consecutive number contains queued messages. BOOLEAN write-read access.

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

LineWidth property (before WinCC V7)

Description
Specifies the line width of the trend referenced by "Index". Value range from 0 to 10.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

LineWidth property

Line weight of window dividers - LineWidth
Defines the line weight of the window dividers in pixels.
The attribute can be assigned dynamic properties by means of the name LineWidth. The data type is LONG.

ListType Property

Description
Returns the data type displayed in the case of a text list object. Read only access.
Value range from 0 to 2.
0 = decimal
1 = binary
2 = bit
LoadDataImmediately property

Load archive data - LoadDataImmediately

Defines whether the tag values for the time range to be displayed are loaded from the archives when the picture is called.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Loads archived values on picture calls.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Loads only current values on picture calls.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name LoadDataImmediately. The data type is BOOLEAN.

LoadDataImmediately property (before WinCC V7)

Description

TRUE, when the tag values for the time range to be displayed are loaded from the archives on opening a picture. BOOLEAN write-read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

LocaleID Property

Description

Defines the language to be displayed in the control, e.g. 1031 for German. Write/Read access. The list of language codes is available in the WinCC documentation (Index > Language Code).
See also
WinCC Slider Control (Page 277)
WinCC Gauge Control (Page 254)
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

LocaleSpecificSettings Property

Description
TRUE if a font can be assigned and formatted for each Runtime language. BOOLEAN write-read access.

LockBackColor Property

Description
Defines or returns the background color of the button for a locked measuring point. LONG write/read access.
The LockStatus property must be set to TRUE for the background color to be displayed.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

LockStatus Property

Description
TRUE, when a locked measuring point should be displayed. BOOLEAN write-read access.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)
LockText Property

Description
Defines the label of a button for a locked measuring point. The LockStatus property must be set to TRUE for the label to be displayed.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

LockTextColor Property

Description
Defines or returns the color of the button label for a locked measuring point. LONG write/read access. The LockStatus property must be set to TRUE for the background color to be displayed.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

Logging Property

Description
Returns an object of type "Logging". Logging (read-only)

See also
HMIRuntime Object (Page 118)
Logging Object (Page 122)

LongStrokesBold Property

Description
TRUE, when the long sections of a scale should be displayed in bold face. BOOLEAN write-read access.
See also

Bar (Page 174)
ScreenItem Object (Page 125)

LongStrokesOnly Property

Description

TRUE, when only the long sections of a scale should be displayed. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

LongStrokesSize Property

Description

Defines or returns the length of the axis section in pixels.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

LongStrokesTextEach Property

Description

Returns the value which defines which sections of the scale displayed should be labeled (1 = every section, 2 = every second section, etc.). Read only access

See also

Bar (Page 174)
ScreenItem Object (Page 125)
LongTimeArchiveConsistency Property

LongTimeArchiveConsistency

If "LongTimeArchiveConsistency" is set to "No", 1000 messages are displayed in the long-term archive list on the single-user system, server or client for each server, or for each redundant server pair.

If the "LongTimeArchiveConsistency" is set to "yes", the most recent 1000 messages are displayed on the client of all servers or redundant server pair in the long-term archive list.

The attribute can be assigned dynamic properties by means of the name LongTimeArchiveConsistency. The data type is BOOLEAN.

LongTimeArchiveConsistency property (before WinCC V7)

Description

If "LongTimeArchiveConsistency" is set to "No", 1000 messages are displayed in the long-term archive list in the single-user system, server or client for each server or for each redundant server pair.

If the "LongTimeArchiveConsistency" is set to "yes", the most recent 1000 messages are displayed on the client of all servers or redundant server pair in the long-term archive list.

Write/Read access.

LowerLimit Property

Description

WinCC Online Trend Control/WinCC Function Trend Control

TRUE, when the "LowerLimitColor" specification is to be used in order to identify the tag values (from a trend referenced via "Index") which lie below the value defined in "LowerLimitValue". BOOLEAN write-read access.

WinCC Online Trend Control

The value of this attribute cannot be changed. Read only access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
LowerLimitColor Property

Description

WinCC Online Trend Control/WinCC Function Trend Control
Defines the color to be used in order to identify the tag values (from trend referenced via "Index") which lie below the value defined in "LowerLimitValue". Whether the information is evaluated is dependent on the value of the "LowerLimit" property. The color is defined as an RGB value. LONG write-read access.

Online Table Control
The value of this attribute cannot be changed. Read only access.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

LowerLimitTagName Property

Description

This defines the lower limit of the trend range, which is automatically taken from the variable properties configured in PCS 7. Write/Read access.

LowerLimitValue Property

Description

WinCC Online Trend Control/WinCC Function Trend Control
Tag values (from a trend referenced via "Index") which lie below the value defined by "LowerLimitValue" are identified by the color specified in "LowerLimitColor". Whether the information is evaluated is dependent on the value of the "LowerLimit" attribute.

Online Table Control
The value of this attribute cannot be changed. Read only access.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Online Table Control (before WinCC V7) (Page 292)
1.14.4.12 M

Ma - Mc

Marker Property

Description

TRUE, when the limit values should be displayed as scale values. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

Max Property

Description

Defines or returns the absolute value in the case of a full value display. This value is displayed if the scale display is active.

See also

Bar (Page 174)
Slider (Page 212)
3D Bar (Page 169)
ScreenItem Object (Page 125)

MaximizeButton Property

Description

TRUE, when the object can be maximized in Runtime. Read only access.
See also

Picture Window (Page 179)
Application Window (Page 173)
ScreenItem Object (Page 125)

MCGUBackColorOff Eigenschaft

Description
Defines or returns the background color for flash status "Off" in the case of the "Departed Unacknowledged" status. LONG write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

MCGUBackColorOn Property

Description
Defines or returns the background color for flash status "On" in the case of the "Departed Unacknowledged" status. LONG write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

MCGUBackFlash Property

Description
TRUE, when the background should flash when a message departs unacknowledged. BOOLEAN write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)
MCGUTTextColorOff Property

Description
Defines or returns the color of the text for flash status "Off" in the case of the "Departed Unacknowledged" status. LONG write-read access.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

MCGUTTextColorOn Property

Description
Defines or returns the background color of the text for flash status "Off" in the case of the "Departed Unacknowledged" status. LONG write-read access.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

MCGUTTextFlash Property

Description
TRUE, when the font should flash when a message departs unacknowledged. BOOLEAN write-read access.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

MCKOBackColorOff Property

Description
Defines or returns the background color for flash status "Off" in the case of the "Arrived" status. LONG write-read access.
See also

Group Display (Page 194)
ScreenItem Object (Page 125)

MCKOBackColorOn Property

Description
Defines or returns the background color for flash status "On" in the case of the "Arrived" status. LONG write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

MCKOBackFlash Property

Description
TRUE, when the background should flash when a message arrives. BOOLEAN write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

MCKOTextColorOff Property

Description
Defines or returns the color of the text for flash status "Off" in the case of the "Arrived" status. LONG write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)
MCKOTextColorOn Property

**Description**
Defines or returns the background color of the text for flash status "On" in the case of the "Arrived" status. LONG write-read access.

**See also**
- Group Display (Page 194)
- ScreenItem Object (Page 125)

MCKOTextFlash Property

**Description**
TRUE, when the font should flash when a message arrives. BOOLEAN write-read access.

**See also**
- Group Display (Page 194)
- ScreenItem Object (Page 125)

MCKQBackColorOff Property

**Description**
Defines or returns the background color for flash status "Off" in the case of the "Departed Acknowledged" status. LONG write-read access.

**See also**
- Group Display (Page 194)
- ScreenItem Object (Page 125)

MCKQBackColorOn Property

**Description**
Defines or returns the background color for flash status "On" in the case of the "Departed Acknowledged" status. LONG write-read access.
See also

Group Display (Page 194)  
ScreenItem Object (Page 125)

MCKQBackFlash Property

Description

TRUE, when the background should flash when a message departs acknowledged. BOOLEAN write-read access.

See also

Group Display (Page 194)  
ScreenItem Object (Page 125)

MCKQTextColorOff Property

Description

Defines or returns the color of the text for flash status "Off" in the case of the "Departed Acknowledged" status. LONG write-read access.

See also

Group Display (Page 194)  
ScreenItem Object (Page 125)

MCKQTextColorOn Property

Description

Defines or returns the background color of the text for flash status "On" in the case of the "Departed Acknowledged" status. LONG write-read access.

See also

Group Display (Page 194)  
ScreenItem Object (Page 125)
MCKQTextFlash Property

Description
TRUE, when the font should flash when a message departs acknowledged. BOOLEAN write-read access.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

MCText Property

Description
Defines or returns the label for the respective message class.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

Me

MeasurePoints Property

Description
The "Index" property references a trend. "MeasurePoints" defines the number of measuring points to be displayed. The information is only evaluated when the "TimeAxis" property is set to the value "-1".

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)
MenuToolBarConfig Property

Description
Loads the given configuration file with configured menu and toolbars or returns the name of the configuration file. STRING (write-read access)

See also
- Picture Window (Page 179)
- HMIRuntime Object (Page 118)

MessageBlockAlign property

Alignment - MessageBlockAlign
Aligns the contents of a selected message block in the table.

To change the alignment, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Left</td>
<td>Aligns the contents of a selected message block to the left.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>Aligns the contents of a selected message block to the center.</td>
</tr>
<tr>
<td>2</td>
<td>Right</td>
<td>Aligns the contents of a selected message block to the right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockAlign`. The data type is LONG.

MessageBlockAutoPrecisions property

Automatic decimal places - MessageBlockAutoPrecisions
Enables automatic setting of the number of decimal places.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The number of decimal places is set automatically. The value in the &quot;Decimal places&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value in the &quot;Decimal places&quot; field is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockAutoPrecisions`. The data type is BOOLEAN.
MessageBlockCaption property

Label - MessageBlockCaption
Defines the label of the column title in the message window for the selected message block. The label specified is active in all Runtime languages.

To change the label, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The attribute can be assigned dynamic properties by means of the name MessageBlockCaption. The data type is STRING.

MessageBlockCount property

MessageBlockCount
Defines the number of message blocks which are available for the message list and the hitlist.

The attribute can be assigned dynamic properties by means of the name MessageBlockCount. The data type is LONG.

MessageBlockDateFormat property

Date format - MessageBlockDateFormat
Defines the date format for displaying messages.

To change the date format, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The following date formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The date format is set automatically.</td>
</tr>
<tr>
<td>dd.MM.yy</td>
<td>Day.Month.Year, e.g. 24.12.07.</td>
</tr>
<tr>
<td>dd.MM.yyyy</td>
<td>Day.Month.Year, e.g. 24.12.2007.</td>
</tr>
<tr>
<td>dd/MM/yy</td>
<td>Day/Month/Year, e.g. 24/12/07.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/2007.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name MessageBlockDateFormat. The data type is STRING.
MessageBlockExponentialFormat property

Exponential notation - MessageBlockExponentialFormat

Specifies the exponential notation for visualization of the values of a selected message block.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The values are displayed with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The values are displayed with decimal notation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockExponentialFormat`. The data type is BOOLEAN.

MessageBlockFlashMode property

Flash mode - MessageBlockFlashMode

Specifies how the content of the selected message block flashes in Runtime when a message appears. The "Flashing on" option must be selected.

To change the setting, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard</td>
<td>The text color switches between the standard color and the flash color when flashing</td>
</tr>
<tr>
<td>1</td>
<td>Switch back-ground color/text color</td>
<td>The color of the background and the text color switch during flashing. You configure the message colors for the type of message in the alarm logging editor.</td>
</tr>
<tr>
<td>2</td>
<td>Switch message color/table color</td>
<td>The message colors and the configured table colors switch during flashing. You configure the message colors for the type of message in the alarm logging editor. Set the table colors in the &quot;Layout&quot; tab in the AlarmControl.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockFlashMode`. The data type is LONG.

MessageBlockFlashOn property

Flashing on - MessageBlockFlashOn

Enables flashing of the selected message block in Runtime after a message was activated.

To change the setting, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Flashing message block content.</td>
</tr>
<tr>
<td>FALSE</td>
<td>No flashing message block content.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `MessageBlockFlashOn`. The data type is BOOLEAN.

**MessageBlockHideText property**

**Content as text - MessageBlockHideText**

Enables the textual display of the content of a selected message block.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockHideText`. The data type is BOOLEAN.

**MessageBlockHideTitleText property**

**Title as text - MessageBlockHideTitleText**

Enables the display of the header of a selected message block in text format.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is not displayed in text format. The option is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is displayed in text format. The option is enabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockHideTitleText`. The data type is BOOLEAN.

**MessageBlockId property**

**MessageBlockId**

Default assignment of the ID number and message block in WinCC AlarmControl.

The attribute can be assigned dynamic properties by means of the name `MessageBlockID`. The data type is LONG.
**MessageBlockInvertUseMessageColor property**

Specifies for the message block whether or not the message colors are displayed, contrary to the central setting for the AlarmControl. For example, the "UseMessageColor" property is set to "FALSE" for the AlarmControl. You have set the "MessageBlockInvertUseMessageColor" property to "TRUE" for a message block. This causes the message colors to be displayed for this message block in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Contrary to the central setting in &quot;UseMessageColor&quot;, the message colors are displayed or not displayed for the message block.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Just like the central setting in &quot;UseMessageColor&quot;, the message colors are displayed or not displayed for the message block.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageBlockInvertUseMessageColor`. The data type is BOOLEAN.

**MessageBlockIndex property**

References an existing message block. Using this attribute, you can assign a specific message block values for other attributes.

Values between 0 and "MessageBlockCount" minus 1 are valid for "MessageBlockIndex". Attribute "MessageBlockCount" defines the number of available message blocks.

The attribute can be assigned dynamic properties by means of the name `MessageBlockIndex`. The data type is LONG.

**MessageBlockLeadingZeros property**

**Number of digits - MessageBlockLeadingZeros**

Defines the number of leading zeros for the message block content. The maximum number is "11". A "0" value deactivates the "With leading zeros" option.

To change the setting, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The attribute can be assigned dynamic properties by means of the name `MessageBlockLeadingZeros`. The data type is LONG.
MessageBlockLength property

Length in characters - MessageBlockLength

Defines the length of the message block selected based on the number of characters.

To change the length, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The attribute can be assigned dynamic properties by means of the name MessageBlockLength. The data type is LONG.

MessageBlockName property

Object name - MessageBlockName

Displays the object name of the message block selected. You cannot edit this name.

The data type is STRING.

MessageBlockPrecisions property

Decimal places - MessageBlockPrecisions

Specifies the decimal precision of the values of a selected message block. You can only enter the value if the "Automatic" option is disabled.

The attribute can be assigned dynamic properties by means of the name MessageBlockPrecisions. The data type is SHORT.

MessageBlockSelected property

Available message blocks - MessageBlockSelected

The available message blocks are blocks that can be used in Runtime for the message list or hitlist.

Select the "Message blocks" tab to activate existing message blocks as required in the Control. Select the "Hitlist" and "Message list" tabs to configure the hitlist and message list based on the available blocks.

To change the setting, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The attribute can be assigned dynamic properties by means of the name MessageBlockSelected. The data type is BOOLEAN.
MessageBlockShowDate property

Show date - MessageBlockShowDate

Enables the display of a date in the "Time" message block in addition to the time.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Date and time are displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time is displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name MessageBlockShowDate. The data type is BOOLEAN.

MessageBlockShowIcon property

Content as icon - MessageBlockShowIcon

Enables the display of the content of a selected message block as icon.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is visualized as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is not visualized as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name MessageBlockShowIcon. The data type is BOOLEAN.

MessageBlockShowTitleIcon property

Title as icon - MessageBlockShowTitleIcon

Enables the display of the title of a selected message block as icon.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is displayed as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is not displayed as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name MessageBlockShowTitleIcon. The data type is BOOLEAN.

MessageBlockTextId property

Text ID - MessageBlockTextId

Specifies the caption of the selected message block using a Text ID which was derived from the text library. The caption is adapted automatically if a user changes the Runtime language.
To change the setting, the option “Apply project settings” must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The attribute can be assigned dynamic properties by means of the name `MessageBlockTextId`. The data type is LONG.

**MessageBlockTimeFormat property**

**MessageBlockTimeFormat**

Defines which time format or duration format is used for displaying the messages.

To change the setting, the option "Apply project settings" must be deactivated or "ApplyProjectSettings" must be set to "FALSE".

The following time formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time format is set automatically.</td>
</tr>
<tr>
<td>HH:mm:ss</td>
<td>Hours:Minutes:Seconds, e.g. 15:35:44</td>
</tr>
<tr>
<td>HH:mm:ss.ms</td>
<td>Hours:Minutes:Seconds.Milliseconds, e.g. 15:35:44.240.</td>
</tr>
<tr>
<td>hh:mm:ss tt</td>
<td>Hours:Minutes:Seconds AM/PM, e.g. 03:35:44:24 PM.</td>
</tr>
<tr>
<td>hh:mm:ss.ms tt</td>
<td>Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.</td>
</tr>
</tbody>
</table>

The following time duration formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time duration format is determined automatically.</td>
</tr>
<tr>
<td>d H:mm:ss</td>
<td>Day Hours:Minutes:Seconds, e.g. 1 2:03:55.</td>
</tr>
<tr>
<td>H:mm:ss.</td>
<td>Hours:Minutes:Seconds, e.g. 26:03:55.</td>
</tr>
<tr>
<td>m:ss</td>
<td>Minutes:Seconds, Example: 1563:55.</td>
</tr>
<tr>
<td>s</td>
<td>Seconds, e.g. 93835.</td>
</tr>
</tbody>
</table>

The attribute can be made dynamic by means of the name `MessageBlockTimeFormat`. The data type is STRING.

**MessageBlockType property**

**MessageBlockType**

Displays the association of the message block.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System block</td>
<td>The message block belongs to the system block category.</td>
</tr>
<tr>
<td>1</td>
<td>Text block</td>
<td>The message block belongs to the user text block category.</td>
</tr>
<tr>
<td>2</td>
<td>Process value block</td>
<td>The message block belongs to the process value block category.</td>
</tr>
<tr>
<td>3</td>
<td>Hitlist block</td>
<td>The message block belongs to the message blocks of the hitlist.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `MessageBlockType`. The data type is LONG.

**MessageClass Property**

**Description**
Defines the respective message type (Alarm High, Alarm Low, Warning High, Warning Low, ...) for which the "Display Text", "Arrived-", "Arrived Acknowledged -" and "Departed Unacknowledged -" settings have been configured.

**See also**
Group Display (Page 194)
ScreenItem Object (Page 125)

**MessageColumnAdd property**

**MessageColumnAdd**
Adds the selected message block from the list of existing message blocks to the list of selected message blocks.

The attribute can be assigned dynamic properties by means of the name `MessageColumnAdd`. The data type is STRING.

**MessageColumnAutoAdjust property**

**MessageColumnAutoAdjust**
Automatically optimizes the width of the columns in the WinCC AlarmControl in Runtime. If the configured column is not sufficient for the displayed text, the column width is increased.

If the user changes the width of columns in Runtime, the width is only optimized again when the picture is opened again.

The attribute can be assigned dynamic properties using the name `MessageColumnAutoAdjust`. The data type is BOOLEAN.

**See also**
MessageColumn object (list) (Page 223)
WinCC AlarmControl (Page 241)
MessageColumnCount property

**MessageColumnCount**

Specifies the number of message blocks to be displayed in the message list in Runtime.

The attribute can be assigned dynamic properties by means of the name `MessageColumnCount`. The data type is LONG.

MessageColumnIndex property

**MessageColumnIndex**

References a message block selected for the message list. Using this attribute you can assign the values of other attributes to a specific message block of the message list.

Values between 0 and "MessageColumnCount" minus 1 are valid for "MessageColumnIndex". Attribute "MessageColumnCount" defines the number of message blocks selected for the message list.

The "MessageColumnIndex" attribute can be assigned dynamic properties by means of attribute `MessageColumnRepos`. The data type is LONG.

MessageColumnName property

**MessageColumnName**

Displays the name of the message block of the message list which is referenced with attribute "MessageColumnIndex". You cannot edit this name.

The attribute can be assigned dynamic properties with the name `MessageColumnName`. The data type is STRING.

MessageColumnRemove property

**MessageColumnRemove**

Cuts the marked message block from the list of selected message blocks and pastes it to the list of available message blocks.

The attribute can be assigned dynamic properties by means of the name `MessageColumnRemove`. The data type is STRING.
MessageColumnRepos property

**Up/Down - MessageColumnRepos/HitlistColumnRepos**

Resorts the message blocks. The "Up" and "Down" commands move the selected message block accordingly in the list. This moves the message block in Runtime Control towards the front or towards the back.

The attribute for the hitlist can be assigned dynamic properties by means of the name HitlistColumnRepos.

The attribute for the message list can be assigned dynamic properties by means of the name MessageColumnRepos.

The data type is LONG.

MessageColumnSort property

**MessageColumnSort**

Defines the sorting order of the message block referenced in "MessageColumnIndex".

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
<td>No sorting</td>
</tr>
<tr>
<td>1</td>
<td>Ascending</td>
<td>Ascending order, starting at the lowest value.</td>
</tr>
<tr>
<td>2</td>
<td>Descending</td>
<td>Descending order, starting at the highest value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name MessageColumnSort. The data type is LONG.

MessageColumnSortIndex property

**MessageColumnSortIndex**

Defines the sorting order of the message block referenced in "MessageColumnIndex". The sorting criterion is removed from "MessageColumnSort" if you set a "0" value.

The attribute can be assigned dynamic properties by means of the name MessageColumnSortIndex. The data type is LONG.

MessageColumnVisible property

**Selected message blocks - MessageColumnVisible/HitlistColumnVisible**

Selected message blocks of message list or hitlist that are displayed in Runtime. Defines whether the message block referenced in "MessageColumnIndex" or "HitlistColumnIndex" is displayed.
The attribute for the message list can be assigned dynamic properties by means of the name `MessageColumnVisible`.

The attribute for the hitlist can be assigned dynamic properties by means of the name `HitlistColumnVisible`.

The data type is BOOLEAN.

**MessageListType property**

**Active list on picture call - MessageListType**

Selection field for defining the active list for picture calls.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Message list</td>
<td>The currently active messages are displayed after a picture was called.</td>
</tr>
<tr>
<td>1</td>
<td>Short-term archive list</td>
<td>A short-term archive list displays the logged messages after the picture was called. The display is updated immediately on activation of new messages.</td>
</tr>
<tr>
<td>2</td>
<td>Long-term archive list</td>
<td>A long-term archive list displays the logged messages after a picture was called.</td>
</tr>
<tr>
<td>3</td>
<td>Lock list</td>
<td>Only the currently locked messages are displayed after a picture was called.</td>
</tr>
<tr>
<td>4</td>
<td>Hitlist</td>
<td>The configured statistics data is displayed after a picture was called.</td>
</tr>
<tr>
<td>5</td>
<td>List of messages to be hidden</td>
<td>The messages to be hidden are displayed at the call of a picture.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `MessageListType`. The data type is LONG.

**Mi - Ms**

**Min Property**

**Description**

Defines or returns the absolute value in the case of the smallest value display. This value is displayed if the scale display is active.

**See also**

Slider (Page 212)

Bar (Page 174)

3D Bar (Page 169)

`ScreenItem Object (Page 125)`
MinuteNeedleHeight Property

Description
Defines or returns the length of the minute hand for the analog clock. The specification of the length is entered as a percentage value in relation to half the length of the short side of the rectangular background. Write/Read access.

Example:
The shorter side of the rectangular background is 100 pixels long. The minute hand length is 80. This results in a length of the minute hand of \((100 \text{ pixels} / 2) \times 0.8 = 40 \text{ pixels}\).

See also
WinCC Digital/Analog Clock (Page 246)  
ScreenItem Object (Page 125)

MinuteNeedleWidth Property

Description
Defines or returns the width of the minute hand for the analog clock. The width is specified as a percentage value related to double the length of the minute hand.

Example:
The length of the minute hand is 40 pixels. The minute hand width is 8. This results in a width of the minute hand of \(40 \text{ pixels} \times 2 \times 0.08 = 6 \text{ pixels}\).

See also
WinCC Digital/Analog Clock (Page 246)  
ScreenItem Object (Page 125)

Moveable Property

Description
TRUE, when the object can be moved in Runtime. Read only access.

See also
Picture Window (Page 179)  
Application Window (Page 173)  
ScreenItem Object (Page 125)
Moveable Property

Movable

Defines whether the control can be moved in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The control can be moved in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The control cannot be moved in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name Moveable. The data type is BOOLEAN.

MsgCtrlFlags Property

Description

Defines the sorting sequence in Alarm Control. Write/Read access.

- 0: The entries are sorted by the value in the time column and in ascending order, i.e. the oldest messages are displayed at the top of the message window.
- 1: The entries are sorted by the value in the time column in descending order, i.e. the oldest messages are displayed at the bottom of the message window. In the case of this value, the "AutoScroll" property is automatically deactivated, otherwise the current message could be moved out of the display area of the message window.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

MsgFilterSQL property (before WinCC V7)

Description

Defines an SQL Statement to the selected messages displayed in the message window. Write/Read access.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)
**MsgFilterSQL property**

**MsgFilterSQL**

Defines one or several SQL statements for the custom selection of messages. Multiple user-defined selections are logically linked by "OR" operation. The SQL statements of "DefaultMsgFilterSQL" and "MsgFilterSQL" are linked logically by "AND" operation if you define a default selection by means of "DefaultMsgFilterSQL".

The attribute can be assigned dynamic properties by means of the name **MsgFilterSQL**. The data type is STRING.

1.14.4.13   N

**Name Property**

**Description of layer and tag object**

Returns the object name. STRING (read only)

- In the case of tags, the name of the tag without server and tag prefix
- In the case of layers, the layer name

**Tags**

The tag "Name" property is used to address the tag via the tag list. The name of a tag can contain a server prefix. In WinCC, tag names are structured according to the following scheme:

<Serverprefix>::<Variablenprefix><Name der Variable>

If the tag name alone is specified, the server prefix and tag prefix are removed from the context of the picture.

If the tag is specified with a server prefix in the tag name, the tags and server prefix of the context are ignored and the server prefix included is used.

**WinCC Function Trend Control Description**

The "Index" property references a trend. "Name" defines the name of the trend.

**Description Project Object**

Returns the name of the current Runtime project. STRING (read only)

**Example:**

The following example returns the name of the current Runtime project as Trace:

```vbscript
'VBS160
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
```
Description of DataItem Object

Returns the name of the DataItem object.

See also

ActiveProject Property (Page 303)
WinCC Function Trend Control (before WinCC V7) (Page 288)
Tag Object (Page 137)
Ellipse segment (Page 147)
Layer Object (Page 120)
DataItem Object (Page 113)

NeedleColor Property

Description

Defines or returns the color of the pointer. LONG write-read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

NormalColor Property

Description

Defines the color of the normal area of the scale. LONG write-read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)
NumberLines Property

Description

Text list
Defines the number of lines the text list object should contain or returns the value. If the amount of configured text is larger than this value, the selection list receives a vertical scroll bar.

Combo box and list box
Defines the number of lines the combo box and list field object should contain or returns the value. You can set a maximum of 32 000 lines.

At the same time, the value of the "Number of rows" attribute specifies the high limit for the "Index" attribute in the "Font" property group. Changing the value can have the following effects:

- Increasing the number: New lines are added at the bottom. The standard labeling of the new field can be changed using the "Text" attribute in the "Font" property group.
- Reducing the number: All lines are removed for which the value of the "Index" attribute is higher than the new number.

See also

Text list (Page 196)
ScreenItem Object (Page 125)

NumItems Property

Description

Returns the number of trends or column pairs (visible and invisible) in the window which have been configured. Write/Read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
Ob - On

Object Property

Description
If a non-WinCC control is used, it is possible that the properties provided by the control have the same names as the general ScreenItem properties. In such cases, the ScreenItem properties have priority. The "hidden" properties of an external control supplier can be accessed using the additional "object" property.

Example:
Address the properties of an external control supplier as follows:
Control.object.type
If the following form alone is used
Control.type
the properties of the ScreenItem object are used in the case of identical names.

See also
Controls (Page 218)
ScreenItem Object (Page 125)

ObjectName Property

Description
Returns the object name.
- In the case of graphic objects, the object name
- In the case of pictures, the picture name
STRING (read only)
Example:

The following example issues the names of all the objects contained in the picture "NewPDL1":

```vbs
'DBS80
Dim objScreen
Dim lngIndex
Dim lngAnswer
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
lngAnswer = MsgBox("Name of object " & lngIndex & ": " & strName, vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
```

**Pictures**

Establish the picture name directly from the "ObjectName" property:

```vbs
'DBS81
MsgBox "Screenname: " & HMIRuntime.ActiveScreen.ObjectName
```

See also

- Screen Object (Page 131)
- ScreenItem Object (Page 125)

**ObjectSizeDeclutteringEnable Property**

**Description**

Returns the ObjectSizeDecluttering properties of a picture.

Upon activated ObjectSizeDecluttering, only objects within a set size range are displayed.

You specify the upper and lower limits for the display range in Graphics Designer under "Tools> Settings > Show/Hide".

BOOLEAN Read-only access.
Example:

The example outputs the Decluttering Properties of the picture NewPDL1 as a trace.

'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine

See also

Screen Object (Page 131)

ObjectSizeDeclutteringMax Property

Description

Using the ObjectSizeDeclutteringMax property, the upper size range of a picture may be read. Objects which are larger than the stated pixel size are no longer displayed when ObjectSizeDecluttering is activated.

LONG read-only access.

Example:

The example outputs the Decluttering Properties of the picture NewPDL1 as a trace.

'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine

See also

Screen Object (Page 131)
ObjectSizeDeclutteringMin Property

Description
Using the ObjectSizeDeclutteringMin property, the lower size range of a picture may be read.
Objects which are smaller than the stated pixel size are no longer displayed when
ObjectSizeDecluttering is activated.
LONG read-only access.

Example:
The example outputs the Decluttering Properties of the picture NewPDL1 as a trace.

'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine

See also
Screen Object (Page 131)

OffsetLeft Property

Description
Defines or returns the distance of the picture from the left edge of the picture window.
The picture is displayed as a cutout of the picture window. The picture scroll bars are located
at the left and upper edge of the picture. If you wish to display the picture in the picture window
by using the horizontal and vertical positioning of the picture scroll bars, use the properties
"ScrollPositionX" and "ScrollPositionY“ for such positioning.

See also
ScrollPositionY Property (Page 562)
ScrollPositionX Property (Page 562)
Picture Window (Page 179)
ScreenItem Object (Page 125)
OffsetTop Property

Description

Defines or returns the distance of the picture from the top edge of the picture window.
The picture is displayed as a cutout of the picture window. The picture scroll bars are located
at the left and upper edge of the picture. If you wish to display the picture in the picture window
by using the horizontal and vertical positioning of the picture scroll bars, use the properties
"ScrollPositionX" and "ScrollPositionY" for such positioning.

See also

ScrollPositionY Property (Page 562)
ScrollPositionX Property (Page 562)
Picture Window (Page 179)
ScreenItem Object (Page 125)

OneY Property

Description

TRUE if only the Y-axis of the trend is displayed in the foreground instead of all Y-axes of the
displayed trends. BOOLEAN write-read access.

Online property (before WinCC V7)

Description

Serves to start or stop updating.

- 0: The updated display is stopped. The values are buffered and updated when the button
  is clicked again.
- -1: The updated display is resumed.

See also

ScreenItem Object (Page 125)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
Online property

Starting refresh - Online

Enables a refresh of displayed values when calling a picture in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the refresh of values on picture calls.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the refresh of values on picture calls.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name Online. The data type is BOOLEAN.

OnTop Property

Description

TRUE, when the object should remain in the foreground in Runtime. Read only access.

See also

Picture Window (Page 179)
Application Window (Page 173)
ScreenItem Object (Page 125)

Op

OperationMessage Property

Description

TRUE, if a message should be output upon successful operation. BOOLEAN Schreib-Lese-Zugriff.

The operation is sent to the message system, and is archived. Using the message system, a message may be output in a message line, for example.

Special features of I/O field, text list and slider

The reason for the operation may only be entered if the "OperationReport" property has been set to TRUE.

See also

Slider (Page 212)
Text list (Page 196)
OperatorMessageID property

**OperatorMessageID**

Default assignment of the ID number and trigger event in WinCC OnlineTableControl:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>EditValue</td>
<td>Trigger event &quot;Change archive value&quot;</td>
</tr>
<tr>
<td>6</td>
<td>InsertValue</td>
<td>Trigger event &quot;Generate archive value&quot;</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageID`. The data type is LONG.

OperatorMessageIndex property

**OperatorMessageIndex**

References the event of an archive value change for an operator message. Using this attribute you can assign the values of other attributes to a specific operator message.

The following values are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Trigger event &quot;Change archive value&quot;</td>
</tr>
<tr>
<td>1</td>
<td>Trigger event &quot;Generate archive value&quot;</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageIndex`. The data type is LONG.

OperatorMessageName property

**Object name - OperatorMessageName**

Displays the name that is referenced with the attribute "OperatorMessageIndex" for message events for operator messages. You cannot edit this name.

The following names are available for message events:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>Message event &quot;Lock&quot;</td>
</tr>
<tr>
<td>Unlock</td>
<td>Message event &quot;Enable&quot;</td>
</tr>
</tbody>
</table>
**Value** | **Explanation**
---|---
Hide | Message event "Hide"
Unhide | Message event "Unhide"
Quit | Message event "Ackn."

The attribute can be assigned dynamic properties by means of the name `OperatorMessageName`. The data type is STRING.

**OperatorMessageNumber property**

**Message number - OperatorMessageNumber**

Define a message number for the selected operator message event if you do not want to use the operator message of WinCC.

The attribute can be assigned dynamic properties by means of the name `OperatorMessageNumber`. The data type is LONG.

**OperatorMessageSelected property**

**Operator messages for - OperatorMessageSelected**

Activate the message events which trigger operator messages in the list.

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSelected`. The data type is BOOLEAN.

**OperatorMessageSource1 property**

**Source - OperatorMessageSource1**

Define the message block of an operated message to be added to "Process value block 1" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 1" of the operator message. Select "1" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource1`. The data type is STRING.

**OperatorMessageSource2 property**

**Source - OperatorMessageSource2**

Define the message block of an operated message to be added to "Process value block 2" of the operator message configured in Source.
An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 2" of the operator message. Select "2" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource2`. The data type is STRING.

**OperatorMessageSource3 property**

**Source - OperatorMessageSource3**

Define the message block of an operated message to be added to "Process value block 3" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 3" of the operator message. Select "3" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource3`. The data type is STRING.

**OperatorMessageSource4 property**

**Source - OperatorMessageSource4**

Define the message block of an operated message to be added to "Process value block 4" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 4" of the operator message. Select "4" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource4`. The data type is STRING.

**OperatorMessageSource5 property**

**Source - OperatorMessageSource5**

Define the message block of an operated message to be added to "Process value block 5" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 5" of the operator message. Select "5" at process value as the message lock of the operated message "User text block 1".
The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource5`. The data type is STRING.

**OperatorMessageSource6 property**

**Source - OperatorMessageSource6**

Define the message block of an operated message to be added to "Process value block 6" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 6" of the operator message. Select "6" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource6`. The data type is STRING.

**OperatorMessageSource7 property**

**Source - OperatorMessageSource7**

Define the message block of an operated message to be added to "Process value block 7" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 7" of the operator message. Select "7" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource7`. The data type is STRING.

**OperatorMessageSource8 property**

**Source - OperatorMessageSource8**

Define the message block of an operated message to be added to "Process value block 8" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 8" of the operator message. Select "8" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource8`. The data type is STRING.
OperatorMessageSource9 property

Source - OperatorMessageSource9

Define the message block of an operated message to be added to "Process value block 9" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 9" of the operator message. Select "9" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name OperatorMessageSource9. The data type is STRING.

OperatorMessageSource10 property

Source - OperatorMessageSource10

Define the message block of an operated message to be added to "Process value block 10" of the operator message configured in Source.

An operator message is to be generated to indicate that a message was locked. The contents of "User text block 1" of the locked message, e.g. "Motor faulty", is to be displayed in "Process value block 10" of the operator message. Select "10" at process value as the message lock of the operated message "User text block 1".

The attribute can be assigned dynamic properties by means of the name OperatorMessageSource10. The data type is STRING.

OperatorMessageSourceType1 property

Transfer as - OperatorMessageSourceType1

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType1. The data type is LONG.

OperatorMessageSourceType2 property

Transfer as - OperatorMessageSourceType2

Specifies the format of the source content for the transfer.
The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource1`. The data type is LONG.

**OperatorMessageSource2 property**

**Transfer as - OperatorMessageSource2**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource2`. The data type is LONG.

**OperatorMessageSource3 property**

**Transfer as - OperatorMessageSource3**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource3`. The data type is LONG.

**OperatorMessageSource4 property**

**Transfer as - OperatorMessageSource4**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSource4`. The data type is LONG.

**OperatorMessageSource5 property**

**Transfer as - OperatorMessageSource5**

Specifies the format of the source content for the transfer.
The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType5`. The data type is LONG.

**OperatorMessageSourceType6 property**

**Transfer as - OperatorMessageSourceType6**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType6`. The data type is LONG.

**OperatorMessageSourceType7 property**

**Transfer as - OperatorMessageSourceType7**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType7`. The data type is LONG.

**OperatorMessageSourceType8 property**

**Transfer as - OperatorMessageSourceType8**

Specifies the format of the source content for the transfer.
The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType8`. The data type is LONG.

**OperatorMessageSourceType9 property**

**Transfer as - OperatorMessageSourceType9**

Defines the format for transferring the source.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source as text.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType9`. The data type is LONG.

**OperatorMessageSourceType10 property**

**Transfer as - OperatorMessageSourceType10**

Specifies the format of the source content for the transfer.

The following formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Text</td>
<td>Transfer the source content in text format.</td>
</tr>
<tr>
<td>1</td>
<td>Value</td>
<td>Transfer the source content as value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `OperatorMessageSourceType10`. The data type is LONG.

**OperationReport Property**

**Description**

TRUE, if the reason for an operation should be recorded. BOOLEAN write/read access. When the object is used or operated in Runtime, a dialog opens in which the operator can input the reason for the operation in the form of text. The operation is sent to the message system, and is archived.
See also

Slider (Page 212)
Text list (Page 196)
I/O Field (Page 184)
ScreenItem Object (Page 125)

Or - Ou

Orientation Property

Description
TRUE, when the text in the object should be displayed horizontally. BOOLEAN write-read access.

Description of the "Connector" object type
Modifies the orientation of the connector. BOOLEAN write-read access.

See also

Connector (Page 167)
Static text (Page 165)
Text list (Page 196)
Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
I/O Field (Page 184)
ScreenItem Object (Page 125)

OuterBevelStyle Property

Description
Defines the 3D effect for the outer bevel of the object.
- 0: No border.
- 1: The border is displayed depressed.
- 2: The border is displayed raised.
- 3: The border is displayed in one color without a 3D effect. The border color is defined by the "BevelColorUp" property.
See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

OuterBevelWidth Property

Description
Defines the width of the outer bevel in pixels.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

Outline Property

Description
TRUE, when the button should be given a black border in addition to the 3D border. BOOLEAN write-read access.

See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

OutputFormat Property

Description
Returns the value for the representation of the output value and sets it. The representation depends on the data format.

See also

I/O Field (Page 184)
ScreenItem Object (Page 125)
OutputValue Property

Description
Determine the default setting for the value to be displayed or returns it. This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.

See also
Text list (Page 196)
I/O Field (Page 184)
ScreenItem Object (Page 125)

1.14.4.15 P

Pa - Pe

PageMode property

Enable paging - PageMode
Enables paging in the long-term archive list. Allows you to display all messages of the short-term archive in the long-term archive list. Use the "Messages per page" or "PageModeMessageNumber" property to determine the number of messages displayed per page.

The page up/down buttons of the toolbar can be used if paging is enabled.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Paging is enabled for the long-term archive list.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Paging is disabled for the long-term archive list.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name PageMode. The data type is BOOLEAN.

PageModeMessageNumber property

Messages per page - PageModeMessageNumber
Defines the number of messages shown per page when paging the long-term archive list.

The attribute can be assigned dynamic properties by means of the name PageModeMessageNumber. The data type is LONG.
Parent Property

Description

Returns a reference to the superordinate object.

Objects within the VBS object model are accessed by hierarchy. You may descend in the picture hierarchy using Screen and ScreenItems. You may ascend in the picture hierarchy by using the Parent property.

Usage

The Parent property can be used as often as required within an object hierarchy. The following section provides a systematic description of how to access all the elements in a hierarchy:

The Command

Dim MyCircle
Set MyCircle = ScreenItem("Circle1")

The Command

MsgBox MyCircle.Parent.Objectname
returns the name of "Picture2" located one layer higher in the object hierarchy than the original ScreenItem object "Circle1".

For example, if you wish to use "Parent" three times, ascend in the object hierarchy by three layers:


returns the name of Picture1.

Reasoning:

- Original reference is to ScreenItem "Circle1"
- "Circle1" is within "Picture2" (Layer 1)
- "Picture2" is within Picture Window2 "BF2" (Layer 2)
- "BF2" is within "Picture 1"(Layer 3)

Example

In the following examples, the object name of the parent object is displayed:

'VBS120
Dim objCircle
Set objCircle = HMIRuntime.Screens("ScreenWindow1").ScreenItems("Circle1")
MsgBox objCircle.Parent.ObjectName

'VBS82
Dim objScrItem
Set objScrItem = HMIRuntime.Screens(1).ScreenItems(1)
MsgBox "Name of BaseScreen: " & objScrItem.Parent.ObjectName

See also

Picture Window (Page 179)
Screen Object (Page 131)
Objects and Lists (Page 107)

PasswordLevel Property

Description

Defines the authorization for operation (e.g. no input or no triggering actions) of the object.
See also

ScreenItem Object (Page 125)

Path Property

Description

Returns the path of the current project (without file name). For a WinCC client without its own path, the path is returned in UNC format, otherwise the local path is returned.

STRING (read access only)

Example:

The following example returns the project path as Trace:

```
'VBS161
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbCrLf
```

See also

Project Object (Page 124)

PercentageAxis property

PercentageAxis

Specifies whether an axis with percentage scaling will be displayed in addition to the value axes in the trend window or diagram window.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The display of an axis with percentage scaling is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The display of an axis with percentage scaling is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `PercentageAxis`. The data type is BOOLEAN.

PercentageAxisAlign property

PercentageAxisAlign

Specifies the alignment of the axis with percentage scaling in the trend window or diagram window.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The axis with percentage scaling is aligned left.</td>
</tr>
<tr>
<td>1</td>
<td>right</td>
<td>The axis with percentage scaling is aligned right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `PercentageAxisAlign`. The data type is LONG.

**PercentageAxisColor property**

Specifies the color of an axis with percentage scaling. The button opens the "Color selection" dialog to select the color.

The attribute can be assigned dynamic properties by means of the name `PercentageAxisColor`. The data type is LONG.

**PersistentRT Property**

**Description**

TRUE, when modified window settings should be retained following a change of picture. Whether the information is evaluated is dependent on the value of the "AllowPersistance" property.

**See also**

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**PersistentRTCS Property**

**Description**

TRUE, when modified settings should be retained following a change of picture and applied in the configuration system. Whether the information is evaluated is dependent on the value of the "AllowPersistance" property. BOOLEAN write-read access.
See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

PersistentRTCSPermission Property

Description

Defines the operator permission which is necessary in order to modify settings related to persistence. The value to be entered must correspond to the number of the requested authorization level in the user administrator. Whether or not the information is to be analyzed depends on the value of the "AllowPersistence" property (does not apply to WinCC Alarm Control).

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

PersistentRTPermission Property

Description

Defines the operator permission which is necessary in order to modify settings related to the persistency in Runtime. The value to be entered must correspond to the number of the requested authorization level in the user administrator. Whether or not the information is to be analyzed depends on the value of the "AllowPersistence" property (does not apply to WinCC Alarm Control).

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
**PicDeactReferenced-Eigenschaft**

**Description**
TRUE, when the picture assigned for the "Disable" status should be saved in the RoundButton object. Otherwise, only the associated object reference is saved. Read only access.

**See also**
Round Button (Page 209)  
ScreenItem Object (Page 125)

**PicDeactTransparent Property**

**Description**
Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Disabled" status should be set to "transparent". LONG Write/Read Access. The color is only set to "Transparent" if the value of the "PicDeactUseTransColor" property is "True".

**See also**
Round Button (Page 209)  
ScreenItem Object (Page 125)

**PicDeactUseTransColor Property**

**Description**
TRUE, when the transparent color defined by the "PicDeactTransparent" property for the "Disable" status should be used. BOOLEAN write-read access.

**See also**
Round Button (Page 209)  
ScreenItem Object (Page 125)
PicDownReferenced Property

Description
TRUE, when the picture assigned for the "On" status is to be saved. Otherwise, only the associated object reference is saved. Read only access.

See also
Round Button (Page 209)
ScreenItem Object (Page 125)

PicDownTransparent Property

Description
Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "On" status should be set to "transparent". LONG Write/Read Access. The color is only set to "Transparent" if the value of the "PicDownUseTransColor" property is "True".

See also
Round Button (Page 209)
ScreenItem Object (Page 125)

PicDownUseTransColor Property

Description
TRUE, when the transparent color defined by the "PicDownTransparent" property for the "On" status should be used. BOOLEAN write-read access.

See also
Round Button (Page 209)
ScreenItem Object (Page 125)

PicReferenced Property

Description
TRUE, when the assigned picture is references the object and is not saved in it. Read only access.
See also

Graphic Object (Page 187)
ScreenItem Object (Page 125)

PictAlignment property

Description
Defines or returns the picture alignment of the picture on the button or round button.
LONG write-read access.

PicTransColor Property

Description
Defines or returns which color of the assigned bitmap object (.bmp, .dib) should be set to "transparent". LONG Write/Read Access.
The color is only set to "Transparent" if the value of the "PicUseTransColor" property is "True".

See also

Graphic Object (Page 187)
ScreenItem Object (Page 125)

Picture Property

Description
Returns the picture name of the background picture for the rectangular background for both the analog and digital clocks. Read only access

See also

WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

PictureBack Property

Description
Returns the picture name of the picture for the object background. Read only access.
PictureDeactivated Property

Description
Defines the picture to be displayed in the "Disable" status or returns the picture name. The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.

See also
Round Button (Page 209)
ScreenItem Object (Page 125)

PictureDirectory property

Directory for pictures (PictureDirectory)
Specifies the name of the subdirectory that is created in the "GraCS" directory of the WinCC project. If pictures are stored in the subdirectory, they are available for the extended status display. If no subdirectory is specified or the subdirectory does not contain any pictures, the pictures in the "GraCS" directory are taken into consideration.

The "Directory for pictures" attribute can be dynamized with the name "PictureDirectory".

PictureDown Property

Description
Defines the picture to be displayed in the "On" status or returns the picture name. The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.

See also
Button (Page 201)
Round Button (Page 209)
ScreenItem Object (Page 125)
PictureName Property

Description

Defines the picture to be displayed in the graphic object in Runtime or returns the picture name. The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.

See also

Graphic Object (Page 187)
ScreenItem Object (Page 125)

PictureSelected Property

Description

Returns the picture name of the picture displayed in the "On" status. "AutoSize" controls the adaptation of the size of picture and buttons. Read only access.

See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

PictureSizeMode property

PictureSizeMode

Specifies the size adjustment between picture and control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Fit size to content</td>
<td>The control is adapted to the picture size.</td>
</tr>
<tr>
<td>1</td>
<td>Fit content to size</td>
<td>The picture is adapted or scaled to the control.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name PictureSizeMode. The data type is LONG.

PictureThumb Property

Description

Returns the picture name of the background picture for the slider. Read only access.
See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

PictureUnselected Property

Description
Returns the picture name of the picture displayed in the "Off" status. "AutoSize" controls the adaptation of the size of picture and buttons. Read only access.

See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

PictureUp Property

Description
Defines the picture to be displayed in the "Off" status or returns the picture name. The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.

See also

Round Button (Page 209)
Button (Page 201)
ScreenItem Object (Page 125)

PicUpReferenced Property

Description
TRUE, when the picture assigned for the "Off" status should be saved in the object. Otherwise, only the associated object reference is saved. Read only access.

See also

Round Button (Page 209)
ScreenItem Object (Page 125)
**PicUpTransparent Property**

**Description**
Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Off" status should be set to "transparent". LONG Write/Read Access. The color is only set to "Transparent" if the value of the "PicUpUseTransColor" property is "True".

**See also**
Round Button (Page 209)
ScreenItem Object (Page 125)

**PicUpUseTransColor Property**

**Description**
TRUE, when the transparent color defined by the "PicUpTransparent" property for "Off" status should be used. BOOLEAN write-read access.

**See also**
Round Button (Page 209)
ScreenItem Object (Page 125)

**PicUseTransColor Property**

**Description**
TRUE, when the transparent color defined by the "PicDeactTransparent" property for the "Disable" status should be used. BOOLEAN write-read access.

**See also**
Graphic Object (Page 187)
ScreenItem Object (Page 125)
PlayEndless property

PlayEndless

Specifies if movies are played endlessly in the control.

The attribute can be assigned dynamic properties by means of the name PlayEndless. The data type is BOOLEAN.

PointCount Property

Description

Defines or returns the number of corner points. Each corner point has position coordinates and is identified via an index.

See also

Polyline (Page 158)
Polygon (Page 156)
ScreenItem Object (Page 125)

Position Property

Description

Defines the presetting for the position of the slider.

This value is used as the start value in Runtime.

To operate the process value linked to this attribute, it is necessary that the process value is also linked to the "Position" event. You will find the event "Position" in the "Event" tab, in the topic tree under SliderCtrl\Property Topics\Control Properties\Value.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)
Precisions Property

Description

WinCC Online Trend Control
The "Index" property references a pair of columns. "Precision" defines the number of decimal places which should be shown in this value column. A maximum of 16 decimal places can be displayed.

WinCC Online Trend Control
Defines the number of decimal places with which the scale value is specified.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

PrecisionX Property

Description

Defines or returns the number of decimal places with which the scale value for the X-axis should be specified. Write/Read access.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

PrecisionY Property

Description

Defines or returns the number of decimal places with which the scale value for the Y-axis should be specified. Write/Read access.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
PredefinedAngles Property

Description
Defines or returns the depth of the display of the 3DBarGraph object. Value range from 0 to 3.
0 = cavalier
1 = isometric
2 = axionometric
3 = freely defined

See also
ScreenItem Object (Page 125)
3D Bar (Page 169)

PreferredTarget property

Preferred picture target (PreferredTarget)
The "Preferred picture target" attribute specifies where the picture change is carried out by the Favorites browser.

| Yes   | The picture change is carried out in this picture screen. In the case of nested picture screens the picture change is carried out at the innermost picture screen with the "Yes" setting. |
| No    | The picture change is carried out in the main screen. |

The "Preferred picture target" attribute can be made dynamic with the name "PreferredTarget".

Pressed Property

Description
TRUE, when the Button or RoundButton object is pressed. BOOLEAN write-read access.

See also
Round Button (Page 209)
ScreenItem Object (Page 125)

PrintBackgroundColor Property

Description
TRUE, if the defined background color is also printed while printing the controls. BOOLEAN write-read access.
PrintJob Property

Description

Defines or reads out which print layout should be used for the printed output.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

PrintJobName property

Current print job view - PrintJobName

Defines the print job triggered by the print function of the "Print" toolbar button. The recommended print job is set for the control by default.

Open the "Select Print Job" dialog using the selection button.

The attribute can be assigned dynamic properties by means of the name PrintJobName. The data type is STRING.

Process Property

Description

Defines or returns presetting for the value to be displayed.
This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.

See also

Slider (Page 212)
Radio box (Page 207)
Check box (Page 204)
Bar (Page 174)
3D Bar (Page 169)
ScreenItem Object (Page 125)
ProcessValue property

Description
Returns an object of type "ProcessValue".

See also
Alarms object (list) (Page 110)

ProjectPath Property

Description
Contains the path and name of the associated project.

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

ProviderClsid Property

Description
The "Index" property references a trend. "ProviderClsid" defines whether an archive tag or an internal or external tag should be displayed in this trend.

- {416A09D2-8B5A-11D2-8B81-006097A45D48}: The trend is connected to an archive tag.
- {A3F69593-8AB0-11D2-A440-00A0C9DBB64E}: The trend is connected to an internal or external tag.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

ProviderType Property

Description
Defines the type of values to be displayed in a trend referenced by "Index". In the case of modification of "ProviderType", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "ProviderType", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".
0: Values are supplied via the API interface.
-1: Display of online or archive tags
-2: Displaying values from a user archive

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

1.14.4.16 Q

QualityCode Property

Description
Defines a standard for the quality of a tag value after being read. The quality code is provided as a 16-bit value for automatic evaluation. After a tag has been written, the value is invalid.
SHORT (read only)

Note
A summary of possible Quality Codes is provided in the WinCC Information System under the heading "Communication" > "Diagnostics" or "Communication" > "Quality Codes".

Example:
The following example indicates the quality of the read value when no errors have occurred during the reading process:

'VBS83
Dim objTag
Dim lngLastErr
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
lngLastErr = objTag.LastError
If 0 = lngLastErr Then
MsgBox objTag.QualityCode
End If
See also

LastError Property (Page 459)
ErrorDescription Property (Page 411)
Tag Object (Page 137)

1.14.4.17 R

Ra - Ri

Radius Property

Description
Defines or returns the radius in pixels.

See also
Pie segment (Page 152)
Circular arc (Page 151)
Circle (Page 149)
Round Button (Page 209)
ScreenItem Object (Page 125)

RadiusHeight Property

Description
Defines or returns the vertical radius in pixels (0 to 5000).

See also
Ellipse segment (Page 147)
Ellipse arc (Page 146)
Ellipse (Page 144)
ScreenItem Object (Page 125)
RadiusWidth Property

Description

Defines or returns the horizontal radius in pixels (0 to 5000).

See also

Ellipse segment (Page 147)
Ellipse arc (Page 146)
Ellipse (Page 144)
ScreenItem Object (Page 125)

RangeMax Property

Description

Defines the maximum absolute value for the value display.
If the "WithLabels" property has the value -1 (yes), this value is displayed on the scale.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

RangeMin Property

Description

Defines the minimum absolute value for the value display.
If the "WithLabels" property has the value -1 (yes), this value is displayed on the scale.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)
Rectangular Property

Description
Defines or returns the side ratio of the rectangular background of the gauge. BOOLEAN write-read access.
FALSE: The size of the gauge can be adjusted to any side ratio by dragging the marking points with the mouse.
TRUE: The size of the gauge can only be adjusted by dragging the marking points with the mouse. The side ratio of the background always remains 1:1.

See also
ScreenItem Object (Page 125)
WinCC Gauge Control (Page 254)

ReferenceRotationLeft Property

Description
Defines or returns the X-coordinate of the reference point about which the object should be rotated in Runtime.
The value of the x coordinate is relative to the object width. Enter the value in percent starting from the left edge of the rectangle enclosing the object.

See also
Line (Page 154)
Polyline (Page 158)
Polygon (Page 156)
ScreenItem Object (Page 125)

ReferenceRotationTop Property

Description
Defines or returns the Y-coordinate of the reference point about which the object should be rotated in Runtime.
The value of the Y-coordinate is relative to the object height. Enter the value in percent starting from the top edge of the rectangle enclosing the object.
See also

ScreenItem Object (Page 125)
Line (Page 154)
Polyline (Page 158)
Polygon (Page 156)

RelayCurves Property

Description
TRUE, when the trends should be displayed staggered. BOOLEAN write-read access.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

Relevant Property

Description
TRUE, when the object will be taken into account when forming the group display. BOOLEAN write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

Replacement Property

Description
The "Index" property references a trend. Values, whose start value is unknown on activating Runtime or for which a substitute value is used, have an unstable status. "Replacement" defines whether such values should be identified by the color defined in "ReplacementColor". BOOLEAN write-read access.
See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ReplacementColor Property

Description

The "Index" property references a trend. Values, whose start value is unknown on activating Runtime or for which a substitute value is used, have an unstable status. "ReplacementColor" defines the color used to identify this value. The color is defined as an RGB value. Whether the information is evaluated is dependent on the value of the "Replacement" property.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

RestrictedCursorMovement property

Limited cursor movement (RestrictedCursorMovement)

The "Limited cursor movement" attribute specifies whether the cursor is fixed on the left-hand side in the I/O field. This corresponds to the default setting when opening the I/O field. As a result, negative values are always displayed correctly You enter the values with decimal places regardless of the cursor position. You no longer have any possibility to change the position during the entry. The decimal place shown in the I/O field is hidden during the entry of a decimal place.

| Yes | The cursor movement is restricted in the I/O field. The cursor is fixed on the left-hand side. |
| No  | The cursor movement is not restricted in the I/O field. |

The "Limited cursor movement" attribute can be made dynamic with the "RestrictedCursorMovement" name.

RightComma Property

Description

Defines or returns the number of decimal places (0 to 20).
Rotation property

**Rotation (Rotation)**

Specifies anticlockwise rotation around the icon center.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The icon is not rotated.</td>
</tr>
<tr>
<td>90</td>
<td>The icon is rotated by 90 degrees.</td>
</tr>
<tr>
<td>180</td>
<td>The icon is rotated by 180 degrees.</td>
</tr>
<tr>
<td>270</td>
<td>The icon is rotated by 270 degrees.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `Rotation`. The data type is LONG.

**RotationAngle Property**

**Description**

**Standard objects**

Defines or returns the rotation angle in degrees.

In Runtime, the object (starting from the configured starting position) is displayed rotated clockwise around the reference point by the specified value. The changed orientation of the object is only visible in Runtime.

The coordinates of the reference point are defined with the "Rotation Reference X" and "Rotation Reference Y" attributes.

**T-piece**

Defines or returns the orientation of a T-piece in degrees.

The attribute can assume one of four values. If you enter another value, it is automatically converted to modulus 360 and rounded up or down to the closest permissible value.

The orientation is produced by rotating the T-piece clockwise around the center point by the specified number of degrees.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The standard position of the T-piece is the shape of the letter &quot;T&quot;</td>
</tr>
<tr>
<td>90</td>
<td>The &quot;leg&quot; of the &quot;T&quot; points towards the left</td>
</tr>
</tbody>
</table>
The "leg" of the "T" points upwards
The "leg" of the "T" points to the right

See also

Line (Page 154)
Polyline (Page 158)
Polygon (Page 156)
ScreenItem Object (Page 125)

RoundCornerHeight Property

Description
Defines or returns the corner radius.
Enter the value as a percentage of half the height of the object.

See also
Rounded rectangle (Page 162)
ScreenItem Object (Page 125)

RoundCornerWidth Property

Description
Defines or returns the corner radius.
Enter the value as a percentage of half the width of the object.

See also
ScreenItem Object (Page 125)

RowNumber property

RowNumber
Specifies the row number of the Row object of a Table Control.
RowScrollbar property

Row scroll bars - RowScrollbar

Enables the display of row scroll bars.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No row scroll bars.</td>
</tr>
<tr>
<td>1</td>
<td>as required</td>
<td>Row scroll bars are displayed if horizontal space requirements of the control are greater than the actually available display area.</td>
</tr>
<tr>
<td>2</td>
<td>always</td>
<td>Row scroll bars are always displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name RowScrollbar. The data type is LONG.

RowTitleAlign property

Row label alignment - RowTitleAlign

Specifies the type of row label alignment.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The row headers are aligned left.</td>
</tr>
<tr>
<td>1</td>
<td>centered</td>
<td>The row headers are aligned to center.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The row headers are aligned right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name RowTitleAlign. The data type is LONG.

RowTitles property

Show row labels - RowTitles

Enables the display of row labels.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The row labels are displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The row labels are not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name RowTitles. The data type is BOOLEAN.
RTPersistence property

Online configuration at the next picture change - RTPersistence

Enables retention of the online configurations of the control after a picture change.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Discard</td>
<td>The current online configurations are discarded at the next picture change.</td>
</tr>
<tr>
<td>1</td>
<td>Retain</td>
<td>The current online configurations are retained at the next picture change.</td>
</tr>
<tr>
<td>2</td>
<td>Reset</td>
<td>All online configurations made are lost. The picture is set to the contents found in the configuration system.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name RTPersistence. The data type is LONG.

RTPersistencePasswordLevel property

Operator authorization for online configuration - RTPersistencePasswordLevel

Displays the authorization for online configuration. You can edit the authorization using the selection button. Authorizations are configured in the "User Administrator" editor.

The attribute can be assigned dynamic properties by means of the name RTPersistencePasswordLevel. The data type is LONG.

RTPersistenceType property

Online configuration - RTPersistenceType

Defines how to retain online configurations of WinCC.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not retain</td>
<td>Online configurations are not retained. These are lost at the next picture change.</td>
</tr>
<tr>
<td>1</td>
<td>Retain during runtime</td>
<td>Online configurations are retained during runtime. These are lost on exiting.</td>
</tr>
<tr>
<td>2</td>
<td>Retain permanently</td>
<td>Online configurations are retained permanently. These are also available after restart.</td>
</tr>
</tbody>
</table>

The attribute cannot be dynamized.
RulerFont Property

Description

This attribute defines the font of the table of the tag values, which is displayed by the key function "Display value at this position" / "Ruler". Write/Read access.

RulerPrecisions Property

Description

Defines the number of decimal places to which a measured value should be displayed when it is determined using the "Display value at this position" function.

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

RulerPrecisionX Property

Description

Defines the number of decimal places used by the "Display value at this position" to display the X-coordinate of a measured value. Whether the information is evaluated is dependent on the value of the "TimeAxisX" attribute.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

RulerPrecisionY Property

Description

Defines the number of decimal places used by the "Display value at this position" to display the Y-coordinate of a measured value.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
RulerType property

Window - RulerType

Specifies window to be displayed during runtime. Depending on the window type, only certain blocks can be used as columns of the WinCC RulerControl.

The following window types can be selected:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;Ruler&quot; window</td>
<td>The ruler window shows the coordinate values of the trends on a ruler or values of a selected row in the table.</td>
</tr>
<tr>
<td>1</td>
<td>&quot;Statistics area&quot; window</td>
<td>The statistics area window shows the values of the low and high limit of trends between two rulers, or displays the selected range in the table.</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Statistics&quot; window</td>
<td>The statistics window shows the statistic evaluation of trends between two rulers, or it displays the selected values in the table.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name RulerType. The data type is LONG.

1.14.4.18 S

Sa - Sc

SameSize Property

Description

TRUE, when all four buttons of a Group Display object have the same size. BOOLEAN write-read access.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

SavedTrend Property

Description

Displays the name of the last saved trend that was exported in WinCC Online Trend Control using the Save Report button. Read only access.
ScaleColor Property

Description
Defines or returns the color of the scale. LONG write-read access.
The "Scaling" property must be set to TRUE for the color to be displayed.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ScaleTicks Property

Description
Defines the number of segments into which the bar will be subdivided by large tick marks of the scale:
0-100: Object can be divided into a maximum of 100 segments
= 0: The optimum number of segments is set automatically.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

Scaling Property

Description
TRUE, when a scale should also be used to represent a value. BOOLEAN write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

ScalingType Property

Description of Bar Scaling
Defines or returns the type of bar scaling. Value range from 0 to 6.
0 = linear
1 = logarithmic
2 = negative logarithmic
3 = automatic (linear)
4 = tangent
5 = square
6 = cubic

The "Scaling" property must be set to TRUE for the color to be displayed.

Description of Online Trend Control

Specifies or returns the type of scaling for a trend referenced by "Index". Value range from 0 to 2.

0 = linear
1 = logarithmic
2 = negative logarithmic

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
Bar (Page 174)
ScreenItem Object (Page 125)

ScalingTypeX Property

Description
Defines the type of scaling of the X-axis of a trend referenced with "Index". Whether the information is evaluated is dependent on the value of the "TimeAxisX" attribute.

0: Linear
-1: Logarithmically. This setting prevents the display of negative values.
-2: Logarithmically negated. This setting prevents the display of positive values.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
ScalingTypeY Property

Description
Defines the type of scaling of the Y-axis of a trend referenced with "Index".
0: Linear
-1: Logarithmically. This setting prevents the display of negative values.
-2: Logarithmically negated. This setting prevents the display of positive values.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ScreenName Property

Description
Defines the picture to be displayed in the picture window in Runtime or returns the picture name.

Note
Always enter picture names without the extension "PDL" for reasons of compatibility with future versions.

See also
Picture Window (Page 179)
ScreenItem Object (Page 125)

Screens Property

Description
Returns an object of type "Screens".
Screens (read only)
Example:

The following example accesses the picture "NewPDL1":

'VBS84
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")

See also

Screens Object (List) (Page 134)
Screen Object (Page 131)
HMIRuntime Object (Page 118)

ScreenItems Property

Description

Returns an object of type "ScreenItems".
ScreenItems (read only)

Example:

The following example issues the number of all the objects contained in the picture "NewPDL1":

'VBS85
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
Msgbox objScreen.ScreenItems.Count

See also

ScreenItems Object (List) (Page 129)
HMIRuntime Object (Page 118)

ScrollBars Property

Description

TRUE, when the object is equipped with a scroll bar in Runtime. Read only access.
See also

Picture Window (Page 179)
ScreenItem Object (Page 125)

ScrollPositionX Property

Description
Specifies the horizontal positioning of the scroll bar in a picture window with slider, or returns its value.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the picture as a cutout where the scroll bars are located at the left and upper edge of the picture, use the properties "OffsetLeft" and "OffsetTop" as the origin of this cutout.

See also

ScreenItem Object (Page 125)
OffsetTop Property (Page 516)
OffsetLeft Property (Page 515)
Picture Window (Page 179)

ScrollPositionY Property

Description
Specifies the vertical positioning of the scroll bar in a picture window with slider, or returns its value.

The picture is displayed in the picture window by positioning the horizontal and vertical scroll bars. If you wish to display the picture as a cutout where the scroll bars are located at the left and upper edge of the picture, use the properties "OffsetLeft" and "OffsetTop" as the origin of this cutout.

See also

OffsetTop Property (Page 516)
OffsetLeft Property (Page 515)
Picture Window (Page 179)
ScreenItem Object (Page 125)
SecondNeedleHeight Property

Description
Defines or returns the length of the second hand for the analog clock. The specification of the length is entered as a percentage value in relation to half the length of the short side of the rectangular background. Write/Read access.

Example:
The shorter side of the rectangular background is 100 pixels long.
The second hand length is 80.
This results in a length of the second hand of (100 pixels / 2) * 0.8 = 40 pixels.

See also
ScreenItem Object (Page 125)
WinCC Digital/Analog Clock (Page 246)

SecondNeedleWidth Property

Description
Defines or returns the width of the second hand for the analog clock. The width is specified as a percentage value related to double the length of the second hand. Write/Read access.

Example:
The length of the second hand is 40 pixels.
The second hand width is 2.
This results in a width of the second hand of 40 pixels * 2 * 0.02 = 2 pixels.

See also
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

SelBGColor Property

Description
Defines or returns the background color of the selected entry in a text list object. LONG write-read access.
See also

Text list (Page 196)
ScreenItem Object (Page 125)

SelectArchiveName property

SelectArchiveName

Opens the dialog for selecting the user archive.

Programmers can set this attribute to allow users to select a user archive by means of a button, for example.

The attribute can be assigned dynamic properties by means of the name SelectArchiveName. The data type is BOOLEAN.

SelectedCellColor property

Background color of selected cell - SelectedCellColor

Specifies the background color of a selected cell. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name SelectedCellColor. The data type is LONG.

SelectedCellForeColor property

Font color of the selected cell - SelectedCellForeColor

Specifies the font color of the selected cell. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name SelectedCellForeColor. The data type is LONG.

SelectedRowColor property

Background color of the selected row - SelectedRowColor

Specifies the background color of the selected line. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name SelectedRowColor. The data type is LONG.
SelectedRowForeColor property

Font color of the selected row - SelectedRowForeColor

Specifies the font color of the selected row. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name SelectedRowForeColor. The data type is LONG.

SelectedTitleColor property

Background selection color - SelectedTitleColor

Specifies the background color of a selected table header. The button opens the "Color selection" dialog.

The setting is only active in Runtime if the "Selection color" or "UseSelectedTitleColor" option is activated.

The attribute can be assigned dynamic properties by means of the name SelectedTitleColor. The data type is LONG.

SelectedTitleForeColor property

Font selection color - SelectedTitleForeColor

Specifies the font color of the table header selected. The button opens the "Color selection" dialog.

The setting is only active in Runtime if the "Selection color" or "UseSelectedTitleColor" option is activated.

The attribute can be assigned dynamic properties by means of the name SelectedTitleForeColor. The data type is LONG.

SelectedTrend Property

Description

This property brings a trend to the foreground via its name. Write/Read access.

SelectionColoring property

Selection colors for - SelectionColoring

Enables the use of selection colors for cells or rows.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>No selection colors for cells and rows.</td>
</tr>
<tr>
<td>1</td>
<td>Cell</td>
<td>Selection color for cell.</td>
</tr>
<tr>
<td>2</td>
<td>Row</td>
<td>Selection color for row.</td>
</tr>
<tr>
<td>3</td>
<td>Cell and row</td>
<td>Selection colors for cell and row.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `SelectionColoring`. The data type is LONG.

### SelectionMode Property

**Description**

Defines whether and how a message line can be selected.

- 0 - NoSelection: Prevents the selection of a message. Acknowledgement affects the oldest pending message.
- 1 - Cell: Enables the selection of fields in the message line. Acknowledgement affects the selected message.
- 2 - Line: Enables the selection of a message line. Acknowledgement affects the selected message.

### See also

- WinCC Alarm Control (before WinCC V7) (Page 286)
- ScreenItem Object (Page 125)

### SelectionRect property

**Selection border- SelectionRect**

Enables the use of a selection border for selected cells or rows.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>No selection border is drawn for selected cells or rows.</td>
</tr>
<tr>
<td>1</td>
<td>Cell</td>
<td>A selection border is drawn for the selected cell.</td>
</tr>
<tr>
<td>2</td>
<td>Row</td>
<td>A selection border is drawn for the selected row.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `SelectionRect`. The data type is LONG.
SelectionRectColor property (before WinCC V7)

Description
Specifies the color of the rectangle in the message window if SelectionType equals "1".

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

SelectionRectColor property

Color of the selection border - SelectionRectColor
Specifies the color of the selection border. The button opens the "Color selection" dialog.
The attribute can be assigned dynamic properties by means of the name SelectionRectColor.
The data type is LONG.

SelectionRectWidth property (before WinCC V7)

Description
Specifies the line weight of the rectangle in the message window if SelectionType equals "1".

See also
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

SelectionRectWidth property

Line weight of the selection border - SelectionRectWidth
Defines the line weight of the selection border in pixels.
The attribute can be assigned dynamic properties by means of the name SelectionRectWidth.
The data type is LONG.
SelectionType property (before WinCC V7)

Description

Specifies if the selected message in the message window should be optically emphasized by color change or rectangle.

- 0 - Color Change: selected message is optically emphasized by color change
- 1 - Rectangle: selected message is optically emphasized by a rectangle

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

Selectable rows - SelectionType

Defines the number of lines you can select. The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>No row selection.</td>
</tr>
<tr>
<td>1</td>
<td>Single selection</td>
<td>One row can be selected.</td>
</tr>
<tr>
<td>2</td>
<td>Multiple selection</td>
<td>Multiple rows can be selected.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name SelectionType. The data type is LONG.

SelIndex property

Description

Defines and returns the index of which the associated text is highlighted in the combobox or list box.

The maximum value is the number of lines (NumberLines) of the object.

SelText property

Description

Shows the text defined with the "Selected field" (SelIndex) attribute which is highlighted in the combobox or list box.
SelTextColor Property

Description
Defines or returns the color of the text of the selected entry in the text list object. LONG write-read access.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

ServerData Property

Description
The attribute can only be modified using the "Properties of WinCC Online Trend Control" dialog. Read only access.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

ServerDataX

Accesses the configured data connection for the X axis with WinCC FunctionTrendControl. The attribute can be assigned dynamic properties by means of the name ServerDataX. The data type is LONG.

Example: Editing the start ID
You may use the ServerDataX attribute to edit the start ID of the X axis.
Prerequisite is that you have an existing trend and trend view, configured X and Y axes, as well as a data connection to the user archive.

In the following example you employ the GetTrend method to set a reference to the object in step one, and then to the trend used in step two. Determine the data connection settings in the third step. Set the start ID to 4 in step 4. The number (3) represents the listing type "user archive" for data transfer. Change the modified data connection settings in step five:
Sub OnCklick(ByVal Item)
1. Step:
   Dim fx_ctrl
   Set fx_ctrl = ScreenItems.Item("Control1")
2. Step:
   Dim fx_trend
   Set fx_trend = fx_ctrl.Getrend("myTrend1")
3. Step:
   Dim vServerDataX, vServerDataY
   vServerDataX = fx_trend.ServerDataX
   vServerDataY = fx_trend.ServerDataY
4. Step:
   Dim startId
   startId = CLng(4)
   vServerDataX(3) = startId
   vServerDataY(3) = startId
5. Step:
   fx_trend.ServerDataX = ServerDataX
   fx_trend.ServerDataY = ServerDataY
End Sub

ServerDataY

Accesses the configured data connection for the Y axis with WinCC FunctionTrendControl.
The attribute can be assigned dynamic properties by means of the name ServerDataY. The data type is LONG.

Example: Editing the start ID

You may use the ServerDataY attribute to edit the start ID of the Y axis.
Prerequisite is that you have an existing trend and trend view, configured X and Y axes, as well as a data connection to the user archive.

In the following example you employ the GetTrend method to set a reference to the object and then to the trend used. Determine the data connection settings in the third step. Set the start ID to 4 in step 4. The number (3) represents the listing type "user archive" for data transfer. Change the modified data connection settings in step five:
Sub OnCklick(ByVal Item)
1. Step:
   Dim fx_ctrl
   Set fx_ctrl = ScreenItems.Item("Control1")
2. Step:
   Dim fx_trend
   Set fx_trend = fx_ctrl.Getrend("myTrend1")

3. Step:
   Dim vServerDataX, vServerDataY
   vServerDataX = fx_trend.ServerDataX
   vServerDataY = fx_trend.ServerDataY

4. Step:
   Dim startId
   startId = CLng(4)
   vServerDataX(3) = startId
   vServerDataY(3) = startId

5. Step:
   fx_trend.ServerDataX = ServerDataX
   fx_trend.ServerDataY = ServerDataY

End Sub

ServerNames property

Server selection - ServerNames

Defines from which servers within a distributed system the message window obtains the display data.

The attribute can be assigned dynamic properties by means of the name ServerNames. The data type is STRING.

ServerNames property (before WinCC V7)

Description

Defines the server in a distributed system to which the data in the message window should relate. Servers are specified as follows: NameServer1;NameServer2;NameServer3. Write/Read access.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

ServerPrefix Property

Description

Defines the server containing the picture to be displayed in the picture window in Runtime or returns the server name.

Enter the server name followed by two colons: "<Servername>::". No check is made as to whether the server actually exists.
See also

Picture Window (Page 179)
ScreenItem Object (Page 125)

Sh - Sk

ShareSpaceWithSourceControl property

Defines whether the size of the source control in the picture window is adapted so that the WinCC RulerControl is also displayed in a small picture window.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The source control in the picture window is adapted.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The source control in the picture window is not adapted.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShareSpaceWithSourceControl`. The data type is BOOLEAN.

ShowBar Property

Description

TRUE, when the bar should be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

ShowBarIcon property

ShowBarIcon

Enables the display of an icon below the value axes. The symbol indicates the diagram currently displayed in the foreground.

The attribute can be assigned dynamic properties using the name `ShowBarIcon`. The data type is BOOLEAN.
ShowDanger Property

Description
Controls the display of the "danger zone" on the instrument scale. BOOLEAN write-read access.
TRUE: The area is identified by the color defined in "DangerColor".
FALSE: The color identification of the area is switched off.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

ShowDecimalPoint Property

Description
TRUE, when the labeling of the scale section should be with decimal numbers (decimal point and one decimal place).
FALSE, when the labeling of the scale section should be with whole numbers.
BOOLEAN write-read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

ShowMilliseconds property

ShowMilliseconds
Specifies whether milliseconds are displayed in the diagnostic buffer view.
The attribute can be assigned dynamic properties using the name ShowMilliseconds. The data type is BOOLEAN.

ShowNormal Property

Description
Controls the display of the "normal zone" on the instrument scale. BOOLEAN write-read access.
TRUE : The area is identified by the color defined for normal color.
FALSE : The color identification of the area is switched off.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

ShowPeak Property

Description
Defines the display of a slave pointer to display the maximum value. BOOLEAN write-read access.
TRUE : The slave pointer is displayed.
FALSE : The slave pointer is hidden.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

ShowPosition Property

Description
TRUE, when the slider position is to be displayed. BOOLEAN write-read access.

See also
WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

ShowRuler property

Show ruler - ShowRuler
Enables the display of a ruler for scanning the coordinate points on picture calls.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of a ruler for scanning the coordinate points.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of a ruler for scanning the coordinate points.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `ShowRuler`. The data type is BOOLEAN.

**ShowRulerImmediately Property**

**Description**

TRUE, when the ruler for determining the coordinate values should be displayed when opening a picture. BOOLEAN write-read access.

**See also**

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**ShowRulerInAxis property**

**ShowRulerInAxis**

Enables the display of rulers in the time axis.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of rulers in the time axes.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of rulers in the time axes.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShowRulerInAxis`. The data type is BOOLEAN.

**ShowScrollbars property**

**Scroll bars - ShowScrollbars**

Enables the display of scroll bars.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>The display of scroll bars is disabled.</td>
</tr>
<tr>
<td>1</td>
<td>as required</td>
<td>Scroll bars are displayed if space requirements of the control are greater than the actual display area.</td>
</tr>
<tr>
<td>2</td>
<td>always</td>
<td>The scroll bars are always displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShowScrollbars`. The data type is LONG.
ShowSlider property

ShowSlider

Specifies if a time slider is displayed in the control.

The attribute can be assigned dynamic properties by means of the name `ShowSlider`. The data type is BOOLEAN.

ShowSortButton property

Use sorting button - ShowSortButton

Enables the display of a sorting button above the vertical scroll bar. Click this sorting button to sort the selected column based on the configured sorting criteria. The sorting button is not displayed if the table does not contain a vertical scroll bar.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables sorting of a selected column by means of sorting button.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The sorting button is not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShowSortButton`. The data type is BOOLEAN.

ShowSortIcon property

Show sorting icon - ShowSortIcon

Enables the display of the sorting icon.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of the sorting icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of the sorting icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShowSortIcon`. The data type is BOOLEAN.

ShowSortIndex property

Show sorting index - ShowSortIndex

Enables the display of a sorting index.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of a sorting index.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of a sorting index.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `ShowSortIndex`. The data type is BOOLEAN.

**ShowSpanNames Property**

**Description**

TRUE, if a section name is also to be displayed in the Value column of Trend Control apart from the measured value and the status display "i" and "u". BOOLEAN write-read access.

**ShowStatisticRuler property**

**ShowStatisticRuler**

Enables the display of rulers in the statistics field on picture calls.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of rulers in the statistics field.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of rulers in the statistics field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ShowStatisticRuler`. The data type is BOOLEAN.

**ShowThumb Property**

**Description**

TRUE, when the slider is to be displayed. BOOLEAN write-read access.

**See also**

WinCC Slider Control (Page 277)

ScreenItem Object (Page 125)

**ShowTitle property**

**Window title - ShowTitle**

Defines representation the Control window header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No window title.</td>
</tr>
<tr>
<td>1</td>
<td>Normal</td>
<td>The window title consists of a WinCC icon and text. The text is entered in the &quot;Text&quot; field.</td>
</tr>
<tr>
<td>2</td>
<td>Narrow</td>
<td>The window title consists only of text. The text is entered in the &quot;Text&quot; field.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `ShowTitle`. The data type is LONG.

**ShowToolbar property**

**ShowToolbar**

Specifies if a toolbar is displayed in the control.

The attribute can be assigned dynamic properties by means of the name `ShowToolbar`. The data type is BOOLEAN.

**ShowTrendIcon property**

**ShowTrendIcon**

Enables the display of an icon below the value axes. The icon indicates the trend currently displayed in the foreground.

The attribute can be assigned dynamic properties by means of the name `ShowTrendIcon`. The data type is BOOLEAN.

**ShowValuesExponentialX Property**

**Description**

TRUE, when the X-coordinate of a measured value determined via the "Display value at this position" function is displayed in exponential notation by a trend referenced via "Index". Whether the information is evaluated is dependent on the value of the "TimeAxisX" property. BOOLEAN write-read access.

**See also**

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**ShowValuesExponentialY Property**

**Description**

TRUE, when the Y-coordinate of a measured value determined via the "Display value at this position" function is displayed in exponential notation by a trend referenced via "Index". BOOLEAN write-read access.
See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ShowWarning Property

Description

Controls the display of the "warning zone" on the instrument scale. BOOLEAN write-read access.
TRUE : The area is identified by the color defined by the warning color attribute.
FALSE : The color identification of the area is switched off.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

SignificantMask Property

Description

Is required in Runtime to display the active message class with the highest priority.
The value of the SignificantMask property represents an internal system output value does not require any specific configuration by the user. Updating is initiated in Runtime by clicking on the object.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

Sizeable property

Sizeable

Enables resizing of the control during runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The control can be resized during runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The control cannot be resized during runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name Sizeable. The data type is BOOLEAN.
SkinName property

Style - SkinName

The control style can be defined in this selection field.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Project setting</td>
<td>The style corresponds to the project settings in WinCC Explorer.</td>
</tr>
<tr>
<td>0</td>
<td>Simple</td>
<td>&quot;Classic&quot; WinCC style</td>
</tr>
<tr>
<td>1</td>
<td>Standard</td>
<td>New WinCC V7 style</td>
</tr>
<tr>
<td></td>
<td>Basic Process Control</td>
<td>The style is reserved for internal use with Basic Process Control.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **SkinName**. The data type is STRING.

**Sm - Sq**

SmallChange Property

**Description**

Defines how many steps the controller can be moved with one mouse click or returns the value.

**See also**

Slider (Page 212)

ScreenItem Object (Page 125)

SmartTag property

**Description**

Returns an object of type "SmartTag".

**See also**

SmartTags Object (Page 136)

SortOrder Property

**Description**

Defines the sort sequence of the message blocks in the message window.
SortSequence property

Sorting order by mouse click - SortSequence

Specifies how to change the sorting order by mouse click.

The following sorting orders are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Up/down/none</td>
<td>You can toggle between ascending, descending and no sorting by means of mouse click.</td>
</tr>
<tr>
<td>1</td>
<td>Up/down</td>
<td>You can toggle between ascending and descending sorting order by means of mouse click.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name SortSequence. The data type is LONG.

SourceBeginTime Property

Description

In the case of online tags and archive tags ("ProviderType" = -1), it defines the starting time of the time range of a trend referenced via "Index" and to be displayed in the trend window. In the case of modification of "SourceBeginTime", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceBeginTime", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceControl property

Source - SourceControl

Defines the control to be interconnected with WinCC RulerControl.

The attribute can be assigned dynamic properties by means of the name SourceControl. The data type is STRING.
SourceControlType property

Type - SourceControlType
Defines the type of control that is interconnected with the WinCC RulerControl in the "Source" field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>The RulerControl is not connected to any source.</td>
</tr>
<tr>
<td>1</td>
<td>OnlineTrend Control</td>
<td>The RulerControl is connected with an OnlineTrendControl.</td>
</tr>
<tr>
<td>2</td>
<td>OnlineTable Control</td>
<td>The RulerControl is connected with an OnlineTableControl.</td>
</tr>
<tr>
<td>3</td>
<td>FunctionTrend Control</td>
<td>The RulerControl is connected with a FunctionTrendControl.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name SourceControlType. The data type is LONG.

SourceEndTime Property

Description
In the case of online tags and archive tags ("ProviderType" = -1), it defines the stopping time of the time range of a trend referenced via "Index" and to be displayed in the trend window. In the case of modification of "SourceEndTime", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceEndTime", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceNumberOfUAValues Property

Description
For values from the user archives ("ProviderType" = -2) it defines the number of values which should be loaded from the user archive for a trend referenced via "Index". In the case of modification of "SourceNumberOfUAValues", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceNumberOfUAValues", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".
See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceNumberOfValues Property

Description
The "Index" property references a trend. In the case of online tags and archive tags
("ProviderType" = -1), "SourceNumberOfValues" defines the number of values which should
be displayed in the trend window. Whether the information is evaluated is dependent on the
value of the "SourceTimeRange" property.

In the case of modification of "SourceNumberOfValues", impermissible combinations with
other attributes for data connection could be created. Therefore, before modifying
"SourceNumberOfValues", the immediate acceptance of the changes must be prevented using
"FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceTagNameX Property

Description
The "Index" property references a trend. In the case of online tags and archive tags
("ProviderType" = -1) "SourceTagNameX" defines the tag which should be displayed along
the X-axis. In the case of modification of "SourceTagNameX", impermissible combinations with
other attributes for data connection could be created. Therefore, before modifying
"SourceTagNameX", the immediate acceptance of the changes must be prevented using
"FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
SourceTagNameY Property

Description
The "Index" property references a trend. In the case of online tags and archive tags ("ProviderType" = -1) "SourceTagNameY" defines the tag which should be displayed along the X-axis. In the case of modification of "SourceTagNameY", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceTagNameY", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceTagProviderDataX Property

Description
The attribute can only be modified using the "Properties of WinCC Function Trend Control" dialog.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceTagProviderDataY Property

Description
The attribute can only be modified using the "Properties of WinCC Function Trend Control" dialog.

See also
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
SourceTimeRange Property

Description

The "Index" property references a trend. In the case of online tags and archive tags ("ProviderType" = -1) "SourceTimeRange" defines how the time range to be displayed in the trend window is defined. In the case of modification of "SourceTimeRange", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceTimeRange", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

0: The time range to be displayed is defined by the starting time (SourceBeginTime) and the number of value pairs (SourceNumberOfValues).

-1: The time range to be displayed is defined by the starting time (SourceBeginTime) and stopping time (SourceEndTime).

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceUAArchive Property

Description

The "Index" property references a trend. In the case of values from the user archives ("ProviderType" = -2), "SourceUAArchive" defines the user archive from which the values should be loaded. In the case of modification of "SourceUAArchive", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceUAArchive" the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceUAArchiveStartID Property

Description

The "Index" property references a trend. In the case of values from the user archives ("ProviderType" = -2), "SourceUAArchiveStartID" defines the data record from which the values should be loaded from the user archive. In the case of modification of "SourceUAArchiveStartID", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceUAArchiveStartID", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".
See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceUAColumnX Property

Description

The "Index" property references a trend. In the case of values from the user archives ("ProviderType" = -2), "SourceUAColumnX" defines the column in the user archive from which the values for the X-axis should be loaded. In the case of modification of "SourceUAColumnX", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceUAColumnX", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SourceUAColumnY Property

Description

The "Index" property references a trend. In the case of values from the user archives ("ProviderType" = -2), "SourceUAColumnY" defines the column in the user archive from which the values for the Y-axis should be loaded. In the case of modification of "SourceUAColumnY", impermissible combinations with other attributes for data connection could be created. Therefore, before modifying "SourceUAColumnY", the immediate acceptance of the changes must be prevented using "FreezeProviderConnections".

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

SplittedViewRatio property

SplittedViewRatio

With a split display, this specifies the portion of the "Diagnostic overview" area is shown compared to the "Detail view" area.

The attribute can be assigned dynamic properties using the name SplittedViewRatio. The data type is DOUBLE.
SquareExtent Property

Description
TRUE, when the size of the clock should be adjustable to any side ratio by dragging the marking points with the mouse. BOOLEAN write-read access.

See also
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

St - Sy

StartAngle Property

Description
Defines or returns the start of the object. The information is in counterclockwise direction in degrees, beginning at the 12:00 clock position.

See also
Pie segment (Page 152)
Circular arc (Page 151)
Ellipse segment (Page 147)
Ellipse arc (Page 146)
ScreenItem Object (Page 125)

State property

Description
Returns the status of a message.

The following table shows the possible states of a message:

<table>
<thead>
<tr>
<th>State</th>
<th>Alarm Log Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Came In</td>
</tr>
<tr>
<td>2</td>
<td>Went Out</td>
</tr>
<tr>
<td>5</td>
<td>Came in and comment</td>
</tr>
<tr>
<td>6</td>
<td>Gone and comment</td>
</tr>
</tbody>
</table>
See also

Alarms object (list) (Page 110)

Statusbar Property

Description

TRUE, when the status line is to be displayed. BOOLEAN write-read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

StatusbarAlignment property

Alignment - StatusbarAlignment

Specifies the alignment of the status bar in the control.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Top</td>
<td>The status bar is aligned to the top edge.</td>
</tr>
<tr>
<td>1</td>
<td>Below</td>
<td>The status bar is aligned to the bottom edge.</td>
</tr>
<tr>
<td>2</td>
<td>Left</td>
<td>The status bar is aligned to the left edge.</td>
</tr>
<tr>
<td>3</td>
<td>Right</td>
<td>The status bar is aligned to the right edge.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `StatusbarAlignment`. The data type is LONG.

StatusbarBackColor property

Background color - StatusbarBackColor

Defines the background color of the status bar. The button opens the "Color selection" dialog to select the color.

For the setting to become active, the "Display" or "StatusbarUseBackColor" option must be activated.

The attribute can be assigned dynamic properties by means of the name `StatusbarBackColor`. The data type is LONG.
StatusBarElementAdd property

New - StatusbarElementAdd

Defines a new, user-defined status bar element. The name set by WinCC can be edited in the "Object name" field.

The attribute can be assigned dynamic properties by means of the name StatusbarElementAdd. The data type is STRING.

StatusBarElementAutoSize property

Automatic - StatusbarElementAutoSize

Enables autosizing of the width of a status bar element selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The width of the selected element is set automatically.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The width of the selected element is not set automatically.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name StatusbarElementAutoSize. The data type is BOOLEAN.

StatusBarElementCount property

StatusBarElementCount

Defines the number of configurable status bar elements.

The attribute can be assigned dynamic properties by means of the name StatusbarElementCount. The data type is LONG.

StatusBarElementIconId property

StatusBarElementIconId

Default assignment of the ID number and icon of a status bar element.

The attribute for custom status bar elements can be made assigned dynamic properties by meaus of the name StatusbarElementIconId. The data type is LONG.

StatusBarElementID property

Object ID - StatusbarElementID

Unique ID of the status bar element selected. WinCC assigns this read only ID number.
The attribute can be assigned dynamic properties by means of the name \texttt{StatusbarElementID}. The data type is \texttt{LONG}.

\textbf{StatusbarElementIndex property}

\textbf{StatusbarElementIndex}

References a status bar element. Using this attribute you can assign the values of other attributes to a specific status bar element.

Values between 0 and "StatusbarElementCount" minus 1 are valid for "StatusbarElementIndex". Attribute "StatusbarElementCount" defines the number of configurable status bar elements.

The "StatusbarElementIndex" attribute can be assigned dynamic properties by means of attribute \texttt{StatusbarElementIndex}. The data type is \texttt{LONG}.

\textbf{StatusbarElementName property}

\textbf{Object name - StatusbarElementName}

Displays the object name of the status bar element selected. You can rename the objects of custom status bar elements.

The "StatusbarElementName" attribute can be assigned dynamic properties by means of attribute \texttt{StatusbarElementRename}. The data type is \texttt{STRING}.

\textbf{StatusbarElementRemove property}

\textbf{Remove - StatusbarElementRemove}

Removes the selected status bar element. You can only remove user-defined status bar element from the list.

The attribute can be assigned dynamic properties by means of the name \texttt{StatusbarElementRemove}. The data type is \texttt{STRING}.

\textbf{StatusbarElementRename property}

\textbf{StatusbarElementRename}

 Renames a custom status bar element which is referenced by means of "StatusbarElementIndex" attribute.

The attribute for custom elements can be assigned dynamic properties by means of the name \texttt{StatusbarElementRename}. "StatusbarElementRename" also sets a dynamic attribute "StatusbarElementName". The data type is \texttt{STRING}. 
StatusbarElementRepos property

**Up/Down - StatusbarElementRepos**

Changes the sorting order of button functions. "Up" and "Down" moves the selected status bar element up or down in the list. This moves the status bar element of the Control towards the front or towards the back in Runtime.

The attribute can be assigned dynamic properties by means of the name `StatusbarElementRepos`. The data type is LONG.

StatusbarElementText property

**StatusbarElementText**

Defines the text to be displayed for the status bar element. You can edit the "StatusbarElementText" attribute for custom elements.

The attribute for custom elements can be assigned dynamic properties by means of the name `StatusbarElementText`. The data type is STRING.

StatusbarElementTooltipText property

**StatusbarElementTooltipText**

Defines the tooltip text for the custom status bar element.

The attribute can be assigned dynamic properties by means of the name `StatusbarElementTooltipText`. The data type is STRING.

StatusbarElementVisible property

**Status bar elements - StatusbarElementVisible**

Activate the elements in the list of status bar elements for their display in Runtime.

Click a list entry to adapt the properties, or to change its position in the status bar of the Control by means of the "Up" and "Down" buttons.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The status bar element is displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The status bar element is not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `StatusbarElementVisible`. The data type is BOOLEAN.
**StatusBarElementUserDefined property**

**StatusBarElementUserDefined**

Indicates whether the project engineer has added the status bar element as a new custom element.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The status bar element is user-defined.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The status bar element is defined by the system.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `StatusBarElementUserDefined`. The data type is BOOLEAN.

**StatusBarElementWidth property**

**Width in pixels - StatusBarElementWidth**

Shows the width of the status bar element selected in pixels. You can define the width if the "Automatic" option is not activated.

The attribute can be assigned dynamic properties by means of the name `StatusBarElementWidth`. The data type is LONG.

**StatusBarFontColor property**

**Font color - StatusBarFontColor**

Defines the color of the text in the status bar.

The attribute can be assigned dynamic properties by means of the name `StatusBarFontColor`. The data type is LONG.

**StatusBarPanes Property**

**Description**

Defines the elements to be displayed in the status bar. Write/Read access.

**See also**

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)
StatusbarShowTooltips property

**Tooltips - StatusbarShowTooltips**

Enables the display of tooltips for the status bar elements in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of tooltips.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of tooltips.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `StatusbarShowTooltips`. The data type is BOOLEAN.

Attribute "StatusbarElementTooltipText" defines the tooltip text.

StatusbarText property

**StatusbarText**

Default text in the status bar.

The attribute can be assigned dynamic properties by means of the name `StatusbarText`. The data type is STRING.

StatusbarUseBackColor property

**Display background color - StatusbarUseBackColor**

Sets a background color for the status bar.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of the background color of the status bar.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of a background color for the status bar.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `StatusbarUseBackColor`. The data type is BOOLEAN.

StatusbarVisible property

**Show status bar - StatusbarVisible**

Enables the display of the status bar of a control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of a status bar.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of a status bar.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `StatusBarVisible`. The data type is BOOLEAN.

### StepSeconds property

**StepSeconds**

Specifies the interval for step forward or step backward in movies.

The attribute can be assigned dynamic properties by means of the name `StepSeconds`. The data type is LONG.

### Stretch Property

**Description**

Defines whether the side ratio is retained or adjustable on changing the icon size. BOOLEAN write-read access.

- **FALSE**: The side ratio is retained on changing the icon size.
- **TRUE**: The side ratio of the icon can be adjusted parallel to changing the icon size.

**See also**

HMI Symbol Library (Page 239)
ScreenItem Object (Page 125)

### SymbolAppearance property

**Foreground mode (SymbolAppearance)**

Specifies the appearance of the icon.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Original</td>
<td>The appearance of the icon corresponds to the multi-color representation in the selection of the &quot;Icons&quot; tab.</td>
</tr>
<tr>
<td>1</td>
<td>Shadow</td>
<td>&quot;Black&quot; lines are maintained as contour lines. Elements of the symbols in other colors are displayed as brightness grades of the current foreground color.</td>
</tr>
<tr>
<td>2</td>
<td>Solid</td>
<td>&quot;Black&quot; lines are maintained as contour lines. All icon elements of other colors are assigned the color value of the current foreground color.</td>
</tr>
<tr>
<td>3</td>
<td>Outline</td>
<td>Lines of the color &quot;Black&quot; are maintained as contour lines. All the elements of the symbol in other colors are assigned the color value of the current background color.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `SymbolAppearance`. The data type is LONG.
Ta - Tic

TableColor property

Row background color 1 - TableColor
Defines the background color of the rows. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name TableColor. The data type is LONG.

TableColor2 property

Row background color 2 - TableColor2
Specifies the background color of "Row color 2". The button opens the "Color selection" dialog.

The setting is only active in Runtime if the "Row color 2" or "UseTableColor2" option is activated. The background colors of "Row color 2" and "Row color 1" are used alternately in this case.

The attribute can be assigned dynamic properties by means of the name TableColor2. The data type is LONG.

TableFocusOnButtonCommand Property

Description
Defines whether the focus is set to the table of the control when a button in a script is clicked.

TableForeColor property

Row font color 1 - TableForeColor
Specifies the font color of the rows. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name TableForeColor. The data type is LONG.

TableForeColor2 property

Row font color 2 - TableForeColor2
Specifies the font color of "Row color 2". The button opens the "Color selection" dialog.
The setting is only active in Runtime if the "Row color 2" or "UseTableColor2" option is activated. The font colors of "Row color 2" and "Row color 1" are used alternately in this case. The attribute can be assigned dynamic properties by means of the name \texttt{TableForeColor2}. The data type is LONG.

**TagName Property**

**Description**

The "Index" property references a trend. "TagName" defines the tag linked to this trend. It is specified in the form "Archivname\Variablenname" to display tags in a process value archive or "TasgName" to display an internal or external tag which is not stored in an archive.

**See also**

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

**TagPrefix Property**

**Description**

Defines or returns the tag prefix which is prefixed to all tags contained in the picture window object. In this way, a picture that is embedded in a picture window retains access to its own tags while another accesses other tags.

Modification of the TagPrefix takes effect when a picture is reloaded. When a picture is changed, this occurs automatically; otherwise, the picture name must be reassigned.

The tag prefix can be freely defined, but must match the name of the structure tags.

---

**Note**

**Do not specify an additional tag prefix in the child picture window**

If a picture window is configured in a referenced picture of a picture window, the tag prefix of the parent picture window is applied to the child picture window of the tag prefix. You should not specify an additional tag prefix in the child picture window, because WinCC does not support bundled structure tags. The interpretation "Tag_prefix1.Tag_prefix2.Tag_name" cannot be used for dynamization.

---

**Note**

The TagPrefix property is not available for the controls.
See also

Picture Window (Page 179)
ScreenItem Object (Page 125)

Tags Property

Description

Returns an object of type "Tags".
Tags (read only)

Example:

The following example accesses the tag "Tag1":

'VBS86
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")

See also

Tags Object (List) (Page 140)
HMIRuntime Object (Page 118)

TagProviderClsid Property

Description

The "Index" property references a trend. "TagProviderClsid" defines whether this trend should display an online tag or archived value. The data is only evaluated for online tags and archive tags ("ProviderType" = -1).

{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}: Online tag.
{416A09D2-8B5A-11D2-8B81-006097A45D48}: Values are read from a process value archive or a user archive.

See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
Template Property

Description

Returns the template for displaying the window content of the "Application Window" object. Read only access.

The following templates are possible depending on the property value:

**Window Contents = Global Script**

"GSC diagnostics"
The application window is supplied by applications of the Global Script. The results of the diagnosis system are displayed.

"GSC Runtime"
The application window is supplied by applications of the Global Script. The analysis results regarding characteristics in Runtime are displayed.

**Window Contents = Print Jobs**

"All Jobs”:
The application window is supplied by the logging system. The available reports are displayed as a list.

"All Jobs - Context Menu”:
The application window is supplied by the logging system. The available reports are displayed as a list. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.

"Job Detail View”:
The application window is supplied by the logging system. The available reports are displayed in a selection menu. Detailed information is displayed for the selected report.

"Selected Jobs - Context Menu”:
The application window is supplied by the logging system. The available reports are displayed as a list. This list only contains reports which you have activated the option "Mark for print job list" in the "Print Job Properties" dialog. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.

See also

ScreenItem Object (Page 125)
Application Window (Page 173)

Text Property

Description

Defines or returns the labeling for an object.
See also

Radio box (Page 207)
Check box (Page 204)
Button (Page 201)
Static text (Page 165)
ScreenItem Object (Page 125)

ThumbBackColor Property

Description
Defines the color of the slider.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

TicColor Property

Description
Defines the color of the scale tick marks. LONG write-read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

TicFont Property

Description
Controls the display of the scale division labeling. Read only access.
The following properties can be set:
- Font
- Font Style
- Font Size
- "Strikethrough" effect
- "Underline" effect
See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

TicOffset Property

Description
Defines the diameter of the imaginary circle on which the scale graduation is set. The value is related to the smaller value of the geometric properties Width and Height.
The ends of the main tick marks of the scale graduation point outwards onto this circle.
Value range from 0 to 1.
0: The scale division is in the middle of the graduated scale disk.
1: The diameter of the imaginary circle for the scale tick marks is the smaller value of the geometric properties Width and Height.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

TicTextColor Property

Description
Defines the color of the labeling of the scale tick marks.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

TicTextOffset Property

Description
Defines the diameter of the imaginary circle on which the labeling of the scale tick marks is set. The value is related to the smaller value of the geometric properties Width and Height.
Value range from 0 to 1.
0: The label is in the middle of the graduated scale disk.
1: The diameter of the imaginary circle for the label is the smaller value of the geometric
properties Width and Height. As a result, part of the label can lie outside the object limits and
is, thus, invisible.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

TicWidth Property

Description
Defines the length of the long tick marks for the scaling. The value is related to the half the
smaller value of the geometric properties Width and Height.
The length of the tick marks for fine scaling is 0.5*scale width.
Value range from 0 to end of scale.
0: No scale graduation is available. The division of the scale into ranges is not visible.
Scaling distance: The scaling division ranges from the middle point of the graduated scale disk
to the value defined by the scaling distance.

See also
ScreenItem Object (Page 125)
WinCC Gauge Control (Page 254)

Ticks Property

Description
TRUE, when the numbered face is displayed. BOOLEAN write-read access.

See also
WinCC Digital/Analog Clock (Page 246)
ScreenItem Object (Page 125)

TicksColor Property

Description
Defines or returns the color of the hour markings on the face of the analog clock. LONG write-
read access.
TickStyle Property

Description
This attribute defines the appearance of the scale. Value Range: 0 to 3.
As a result of the automatic scaling, it is possible that, occasionally, two scale tick marks lie
directly beside each other (apparently wide tick mark). This effect can be corrected by minimally
lengthening or shortening the slider object.
It is also possible to completely suppress display of the scaling ("WithAxes").

TimeAxis - TimeBase

TimeAxis Property

Description
Defines whether a common time axis should be used for all trends in the trend window.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)
The attribute can be assigned dynamic properties by means of the name `TimeAxisActualize`. The data type is BOOLEAN.

**TimeAxisAdd property**

**New - TimeAxisAdd**

Creates a new time axis.

The attribute can be assigned dynamic properties by means of the name `TimeAxisAdd`. The data type is STRING.

**TimeAxisAlign property**

**Alignment - TimeAxisAlign**

Specifies the mode of alignment of a selected time axis.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bottom</td>
<td>The selected time axis is displayed below the trend or diagram.</td>
</tr>
<tr>
<td>1</td>
<td>Top</td>
<td>The selected time axis is displayed above the trend or diagram.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeAxisAlign`. The data type is LONG.

**TimeAxisBarWindow property**

**Diagram window - TimeAxisBarWindow**

Specifies the diagram window in which the selected time axis will be used. You can specify the available diagram window in the "Diagram window" tab or using "BarWindowAdd".

The attribute can be assigned dynamic properties using the name `TimeAxisBarWindow`. The data type is STRING.

**TimeAxisBeginTime property**

**Start time - TimeAxisBeginTime**

Defines the start of the time range for a selected time axis.

The attribute can be assigned dynamic properties by means of the name `TimeAxisBeginTime`. The data type is Date.

Use the "yyyy-mm-dd hh:mm:ss" format when setting a dynamic time range.
TimeAxisColor property

Time axis color - TimeAxisColor

Specifies the color of the time axis. The button opens the "Color selection" dialog to select the color.

The setting is only active if the "Use trend color" or "Use diagram color" option is disabled or if "ValueAxisInTrendColor" is "FALSE".

The attribute can be assigned dynamic properties by means of the name TimeAxisColor. The data type is LONG.

TimeAxisCount property

TimeAxisCount

Defines the number of time axes configured.

The attribute can be assigned dynamic properties by means of the name TimeAxisCount. The data type is LONG.

TimeAxisDateFormat property

Date format - TimeAxisDateFormat

Defines the date format for visualizing a selected time axis.

The following date formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The date format is set automatically.</td>
</tr>
<tr>
<td>dd.MM.yy</td>
<td>Day.Month.Year, e.g. 24.12.07.</td>
</tr>
<tr>
<td>dd.MM.yyyy</td>
<td>Day.Month.Year, e.g. 24.12.2007.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/07.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/2007.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeAxisDateFormat. The data type is STRING.

TimeAxisEndTime property

End time - TimeAxisEndTime

Defines the end of the time range of a selected time axis.

The attribute can be assigned dynamic properties by means of the name TimeAxisEndTime. The data type is Date.

Use the "yyyy-mm-dd hh:mm:ss" format when setting a dynamic time range.
TimeAxisFormat Property

Description

Defines the format of the information along the time axis.

- 0: The information is provided in hh:mm
- -1: The information is provided in hh:mm:ss
- -2: The information is provided in hh:mm:ss.ms
- -3: The information is provided in hh:mm (full hours)
- -4: The information is provided in hh:mm:ss (full minutes)
- -5: The information is provided in hh:mm:ss.ms (full seconds)

See also

WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

TimeAxisInBarColor property

In diagram color - TimeAxisInBarColor

Specifies whether the selected time axis will be displayed in the diagram color. If there is more than one diagram in the diagram window, the color of the first diagram will be used. You specify the order of the diagrams on the "Diagrams" tab.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The selected time axis is displayed in the diagram color. The setting in the &quot;Color&quot; or &quot;TimeAxisColor&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The selected time axis is displayed in the color set in the &quot;Color&quot; or &quot;TimeAxisColor&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name TimeAxisInBarColor. The data type is BOOLEAN.

TimeAxisIndex property

References a configured time axis. Using this attribute you can assign the values of other attributes to a specific time axis.

Values between 0 and "TimeAxisCount" minus 1 are valid for "TimeAxisIndex". Attribute "TimeAxisCount" defines the number of trends configured.

The "TimeAxisIndex" attribute can be assigned dynamic properties by means of attribute TimeAxisRepos. The data type is LONG.
TimeAxisInTrendColor property

**Use trend color - TrendAxisInTrendColor**

Sets a trend color for displaying the time axis selected. The color of the first trend is activated if several trends are displayed in the trend window. Define the order of trends on the “Trends” tab.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The trend color is used to display the time axis selected. The setting in the &quot;Color&quot; or &quot;TimeAxisColor&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time axis selected is displayed in the color set in the &quot;Color&quot; or &quot;TimeAxisColor&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TimeAxisInTrendColor**. The data type is BOOLEAN.

TimeAxisLabel property

**Label - TimeAxisLabel**

Defines the label text for a time axis.

The attribute can be assigned dynamic properties by means of the name **TimeAxisLabel**. The data type is STRING.

TimeAxisMeasurePoints property

**Number of measurement points - TimeAxisMeasurePoints**

Defines the number of measurement points to be displayed at the time axis selected.

The attribute can be assigned dynamic properties by means of the name **TimeAxisMeasurePoints**. The data type is LONG.

TimeAxisName property

**Object name - TimeAxisName**

Specifies the name of a selected time axis.

The "TimeAxisName" attribute can be assigned dynamic properties by means of attribute **TimeAxisRename**. The data type is STRING.
**TimeAxisRangeType property**

**Time range setting - TimeAxisRangeType**

Specifies the time range for the time axis selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Time range</td>
<td>Defines the start time and the time range for the time axis.</td>
</tr>
<tr>
<td>1</td>
<td>Start to end time</td>
<td>Defines the start and end time for the time axis.</td>
</tr>
<tr>
<td>2</td>
<td>Number of measurement</td>
<td>Defines the start time and the number of measurement points for the time</td>
</tr>
<tr>
<td></td>
<td>points</td>
<td>axis.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeAxisRangeType`. The data type is LONG.

**TimeAxisRemove property**

**Remove - TimeAxisRemove**

Removes the selected time axis from the list.

The attribute can be assigned dynamic properties by means of the name `TimeAxisRemove`. The data type is STRING.

**TimeAxisRename property**

**TimeAxisRename**

Renames a time axis which is referenced by means of "TimeAxisIndex" attribute.

The attribute can be assigned dynamic properties by means of the name `TimeAxisRename`. "TimeAxisRename" also sets a dynamic attribute "TimeAxisName". The data type is STRING.

**TimeAxisRepos property**

**Up/Down - TimeAxisRepos**

Changes the order of the time axes. "Up" and "Down" move the selected time axis up or down in the list.

The list order determines the time axis position in the trend window or diagram window in runtime. If the orientation is the same and the time axis is further above, the time axis is shown at a more remote position of the curve or diagram.

The attribute can be assigned dynamic properties by means of the name `TimeAxisRepos`. The data type is LONG.
TimeAxisShowDate property

Show date - TimeAxisShowDate

Enables the display of the date and time at the time axis selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Date and time are displayed. The date format is defined in the &quot;Date format&quot; field.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The date is not displayed. Only the time is displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeAxisShowDate. The data type is BOOLEAN.

TimeAxisTimeFormat property

Time format - TimeAxisTimeFormat

Defines the time format for visualizing a selected time axis.

The following time formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time format is set automatically.</td>
</tr>
<tr>
<td>hh:mm:ss.ms</td>
<td>Hours:Minutes:Seconds, e.g. 15:35:44.240.</td>
</tr>
<tr>
<td>hh:mm:ss tt</td>
<td>Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.</td>
</tr>
<tr>
<td>hh:mm:ss.ms tt</td>
<td>Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeAxisTimeFormat. The data type is STRING.

TimeAxisTimeRangeBase property

Time range - TimeAxisTimeRangeBase

Defines the time unit for calculating the time range.

The following time units are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500 ms</td>
</tr>
<tr>
<td>1000</td>
<td>1 second</td>
</tr>
<tr>
<td>60000</td>
<td>1 minute</td>
</tr>
<tr>
<td>360000</td>
<td>1 hour</td>
</tr>
<tr>
<td>8640000</td>
<td>1 day</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeAxisTimeRangeBase. The data type is LONG.
**TimeAxisTimeRangeFactor property**

**Time range - TimeAxisTimeRangeFactor**

Defines the factor for calculating the time range. Only integer factors are valid.

The attribute can be assigned dynamic properties by means of the name `TimeAxisTimeRangeFactor`. The data type is SHORT.

**TimeAxisTrendWindow property**

**Trend window - TimeAxisTrendWindow**

Specifies the trend window for displaying the time axis selected. Define the available trend windows in the "Trend window" or "TrendWindowAdd" tab.

The attribute can be assigned dynamic properties by means of the name `TimeAxisTrendWindow`. The data type is STRING.

**TimeAxisVisible property**

**Time axis - TimeAxisVisible**

The list shows all time axes you created. Click a time axis entry in the list to adapt the properties and to assign a trend window or diagram window to the time axis.

Select the time axes in the list that you want to display in the trend windows or diagram windows.

Defines whether the selected time axis is displayed.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The time axis is displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time axis is not displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeAxisVisible`. The data type is BOOLEAN.

**TimeAxisX Property**

**Description**

TRUE, when a common axis should be used for all trends in the trend window. BOOLEAN write-read access.
See also

WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

TimeBase property

Time base - TimeBase

This selection field is used to define the time base for the time stamp in the control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Local time zone</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC)</td>
</tr>
<tr>
<td>2</td>
<td>Project setting</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeBase`. The data type is LONG.

TimeColumn

TimeColumnActualize property

TimeColumnActualize

Enables the update of values in the selected column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The time column is updated.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The time column is not updated. This setting can be useful when comparing tables.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnActualize`. The data type is BOOLEAN.

TimeColumnAdd property

New - TimeColumnAdd

Creates a new time column.

The attribute can be assigned dynamic properties by means of the name `TimeColumnAdd`. The data type is STRING.
TimeColumnAlign property

Alignment - TimeColumnAlign

Defines the mode of alignment of the time column selected.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The time column selected is displayed on the left.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The time column selected is aligned to center.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The time column selected is displayed on the right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnAlign`. The data type is LONG.

TimeColumnAlignment Property

Description

The "Index" property references a pair of columns. "TimeColumnAlignment" defines the alignment of the time column for this column pair.

- 0: Time values are entered aligned left.
- 1: Time values are entered centered.
- 2: Time values are entered aligned right.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

TimeColumnBackColor property

Background color - TimeColumnBackColor

Specifies the background color of the time column selected. Use the button to open the "Color selection" dialog.

The setting is useful if:

- The "Use value column colors" option is not activated or "TimeColumnUseValueColumnColors" is "FALSE".
- The "Background color" option is activated or "UseColumnBackColor" is "TRUE" in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name `TimeColumnBackColor`. The data type is LONG.
TimeColumnBeginTime property

Start time - TimeColumnBeginTime

Defines the start of the time range for a selected time column.
The attribute can be assigned dynamic properties by means of the name TimeColumnBeginTime. The data type is Date.
Use the "yyyy-mm-dd hh:mm:ss" format when setting a dynamic time range.

TimeColumnCaption property

Caption - TimeColumnCaption

Defines the caption of the time column.
The attribute can be assigned dynamic properties by means of the name TimeColumnCaption. The data type is STRING.

TimeColumnCount property

TimeColumnCount

Defines the number of time columns configured.
The attribute can be assigned dynamic properties by means of the name TimeColumnCount. The data type is LONG.

TimeColumnDateFormat property

Date format - TimeColumnDateFormat

Defines the date format for visualizing a selected time column.
The following date formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The date format is set automatically.</td>
</tr>
<tr>
<td>dd.MM.yy</td>
<td>Day.Month.Year, e.g. 24.12.07.</td>
</tr>
<tr>
<td>dd.MM.yyyy</td>
<td>Day.Month.Year, e.g. 24.12.2007.</td>
</tr>
<tr>
<td>dd/MM/yy</td>
<td>Day/Month/Year, e.g. 24/12/07.</td>
</tr>
<tr>
<td>dd/MM/yyyy</td>
<td>Day/Month/Year, e.g. 24/12/2007.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeColumnDateFormat. The data type is STRING.
**TimeColumnEndTime property**

**End time - TimeColumnEndTime**

Defines the end of the time range of a selected time column.

The attribute can be assigned dynamic properties by means of the name `TimeColumnEndTime`. The data type is Date.

Use the "yyyy-mm-dd hh:mm:ss" format when setting a dynamic time range.

**TimeColumnForeColor property**

**Font color - TimeColumnForeColor**

Specifies the font color of the time column selected. Use the button to open the "Color selection" dialog.

The setting is useful if:

- The "Use value column colors" option is not activated or "TimeColumnUseValueColumnColors" is "FALSE".
- The "Font color" option is activated or "UseColumnForeColor" is "TRUE" in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name `TimeColumnForeColor`. The data type is LONG.

**TimeColumnHideText property**

**TimeColumnHideText**

Sets text format for displaying the content of a time column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is not displayed in text format.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is displayed in text format.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnHideText`. The data type is BOOLEAN.
**TimeColumnHideTitleText property**

**TimeColumnHideTitleText**

Sets text format for displaying the time column header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is not displayed in text format.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is displayed in text format.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnHideTitleText`. The data type is BOOLEAN.

**TimeColumnIndex property**

**TimeColumnIndex**

References a configured time column. Using this attribute you can assign the values of other attributes to a specific time column.

Values between 0 and "TimeColumnCount" minus 1 are valid for "TimeColumnIndex". Attribute "TimeColumnCount" defines the number of time columns configured.

The "TimeColumnIndex" attribute can be assigned dynamic properties by means of attribute `TimeColumnRepos`. The data type is LONG.

**TimeColumnLength property**

**Length in characters - TimeColumnLength**

Specifies the width of a selected time column.

The attribute can be assigned dynamic properties by means of the name `TimeColumnLength`. The data type is LONG.

**TimeColumnMeasurePoints property**

**Number of measurement points - TimeColumnMeasurePoints**

Defines the number of measurement points to be displayed in the time column selected.

The attribute can be assigned dynamic properties by means of the name `TimeColumnMeasurePoints`. The data type is LONG.
TimeColumnName property

Object name - TimeColumnName

Specifies the name of a selected time column.

The "TimeColumnName" attribute can be assigned dynamic properties by means of attribute
TimeColumnRename. The data type is STRING.

TimeColumnRangeType property

Time range setting - TimeColumnRangeType

Defines the time range setting for the time column selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Time range</td>
<td>Defines the start time and time range of the time column.</td>
</tr>
<tr>
<td>1</td>
<td>Start to end time</td>
<td>Defines the start and end time for the time column.</td>
</tr>
<tr>
<td>2</td>
<td>Number of measurement points</td>
<td>Defines the start time and the number of measurement points for the time column.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name
TimeColumnRangeType. The data type is LONG.

TimeColumnRemove property

Remove - TimeColumnRemove

Removes the selected time column from the list.

The attribute can be assigned dynamic properties by means of the name
TimeColumnRemove. The data type is STRING.

TimeColumnRename property

TimeColumnRename

Renames a time column which is referenced by means of "TimeColumnIndex" attribute.

The attribute can be assigned dynamic properties by means of the name
TimeColumnRename. "TimeColumnRename" also sets a dynamic attribute
"TimeColumnName". The data type is STRING.
TimeColumnRepos property

Up/Down - TimeColumnRepos

Repositions the order of time columns and of corresponding value columns. "Up" and "Down" move the time column selected up or down in the list. This moves the time column and corresponding value columns in the table towards the front or towards the back.

The attribute can be assigned dynamic properties by means of the name TimeColumnRepos. The data type is LONG.

TimeColumnShowDate property

Show date - TimeColumnShowDate

Enables the display of the date and time in the time column selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Date and time are displayed. The date format is defined in the &quot;Date format&quot; field or by using &quot;TimeColumnDateFormat&quot;.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The date is not displayed. Only the time is displayed.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeColumnShowDate. The data type is BOOLEAN.

TimeColumnShowIcon property

TimeColumnShowIcon

Enables the display of time column contents as icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is visualized as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is not visualized as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TimeColumnShowIcon. The data type is BOOLEAN.

TimeColumnShowTitleIcon property

TimeColumnShowTitleIcon

Enables display of the time column header as icon. This function is only available in WinCC Alarm Control.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is displayed as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is not displayed as icon.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `TimeColumnShowTitleIcon`. The data type is BOOLEAN.

**TimeColumnSort property**

**TimeColumnSort**

Defines the sorting order of the time column referenced in "TimeColumnIndex".

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No sorting</td>
</tr>
<tr>
<td>1</td>
<td>Ascending</td>
<td>Ascending order, starting at the lowest value.</td>
</tr>
<tr>
<td>2</td>
<td>Descending</td>
<td>Descending order, starting at the highest value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnSort`. The data type is LONG.

**TimeColumnSortIndex property**

**TimeColumnSortIndex**

Defines the sorting order of the time column referenced in "TimeColumnIndex". The sorting criterion is removed from "TimeColumnSort" if you set a "0" value.

The attribute can be assigned dynamic properties by means of the name `TimeColumnSortIndex`. The data type is LONG.

**TimeColumnTimeFormat property**

**Time format - TimeColumnTimeFormat**

Defines the time format for visualizing a selected time column.

The following time formats are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>The time format is set automatically.</td>
</tr>
<tr>
<td>HH:mm:ss.ms</td>
<td>Hours:Minutes:Seconds, e.g. 15:35:44.240.</td>
</tr>
<tr>
<td>hh:mm:ss tt</td>
<td>Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.</td>
</tr>
<tr>
<td>hh:mm:ss ms</td>
<td>Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnTimeFormat`. The data type is STRING.
TimeColumnTimeRangeBase property

Time range - TimeColumnTimeRangeBase

Defines the time unit for calculating the time range.

The following time units are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500 ms</td>
</tr>
<tr>
<td>1000</td>
<td>1 second</td>
</tr>
<tr>
<td>60000</td>
<td>1 minute</td>
</tr>
<tr>
<td>3600000</td>
<td>1 hour</td>
</tr>
<tr>
<td>86400000</td>
<td>1 day</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnTimeRangeBase`. The data type is LONG.

TimeColumnTimeRangeFactor property

Time range - TimeColumnTimeRangeFactor

Defines the factor for calculating the time range. Only integer factors are valid.

The attribute can be assigned dynamic properties by means of the name `TimeColumnTimeRangeFactor`. The data type is SHORT.

TimeColumnUseValueColumnColors property

Use value column colors - TimeColumnUseValueColumnColors

Defines whether the selected time column will be displayed in the value column colors.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The colors of the value column are used to display a selected time column. The settings in the &quot;Font color&quot; and &quot;Background color&quot; fields are disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The colors defined in the &quot;Font color&quot; and &quot;Background color&quot; fields are used to display the selected time column.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TimeColumnUseValueColumnColors`. The data type is BOOLEAN.

TimeColumnVisible property

Time columns - TimeColumnVisible

The list shows the time columns you created. Click a time column entry in the list to adapt the properties and to define the time range of the time column.
Select the time columns to be displayed in the table from the list.
Defines whether the selected time column is displayed.
The attribute can be assigned dynamic properties by means of the name `TimeColumnVisible`. The data type is BOOLEAN.

**TimeFormat - Tolerance**

**TimeFormat Property**

**Description**
Defines the format of the time specification.

- 0: The information is provided in hh:mm
- -1: The information is provided in hh:mm:ss
- -2: The information is provided in hh:mm:ss.ms
- -3: The information is provided in hh:mm (full hours)
- -4: The information is provided in hh:mm:ss (full minutes)
- -5: The information is provided in hh:mm:ss.ms (full seconds)

**See also**
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

**TimeJump Property**

**Description**
WinCC Online Trend Control
The "Index" property references a trend. "TimeJump" defines whether the time jumps in the archive should be identified by the color defined in "TimeJumpColor".

WinCC Online Trend Control
The value of this attribute cannot be changed. Read only access.

**See also**
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)
**TimeJumpColor Property**

**Description**

**WinCC Online Trend Control**

The "Index" property references a trend. "TimeJumpColor" defines the color identifying the time jumps in the archive. Whether the information is evaluated is dependent on the value of the "TimeJump" property. The color is defined as an RGB value. LONG write-read access.

**WinCC Online Trend Control**

The value of this property cannot be changed. Read only access.

**See also**

ScreenItem Object (Page 125)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)

**TimeOverlap Property**

**Description**

**WinCC Online Trend Control**

The "Index" property references a trend. "TimeOverlap" defines whether the time overlaps in the archive should be identified by the color defined in "TimeOverlapColor".

**WinCC Online Trend Control**

The value of this property cannot be changed. Read only access.

**See also**

WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

**TimeOverlapColor Property**

**Description**

**WinCC Online Trend Control**

The "Index" property references a trend. "TimeOverlapColor" defines the color identifying the time overlaps in the archive. Whether the information is evaluated depends on the value of the "TimeOverlap" attribute. The color is defined as an RGB value.
WinCC Online Trend Control
The value of this property cannot be changed. Read only access.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
ScreenItem Object (Page 125)

TimeRange Property

Description
The "Index" property references a column pair or a trend. "TimeRange" defines how the time range to be displayed should be defined.
- 0: The time range to be displayed is defined by a start time ("BeginTime") and end time ("EndTime").
- -1: The time range to be displayed is defined by a start time ("BeginTime") and a time range ("TimeRangeBase" and "TimeRangeFactor").

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

TimeRangeBase Property

Description
The "Index" property references a column pair or a trend. The time range to be displayed for this column pair/trend results from multiplying the values "TimeRangeBase" and "TimeRangeFactor", whereby the value "TimeRangeBase" is interpreted in milliseconds.
The "TimeRangeBase" and "TimeRangeFactor" properties are only evaluated when the "TimeRange" property is set, i.e. has the value "-1".

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)
TimeRangeFactor Property

Description
The "Index" property references a column pair or a trend. The time range to be displayed for this column pair/trend results from multiplying the values "TimeRangeBase" and "TimeRangeFactor", whereby the value "TimeRangeBase" is interpreted in milliseconds.

The "TimeRangeBase" and "TimeRangeFactor" properties are only evaluated when the "TimeRange" property is set, i.e. has the value "-1".

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

TimeStamp Property

Description
Reads the time stamp of the last read access of a tag. The time stamp is returned in local time. DATE (read only)

The VBS standard function "FormatDateTime(Date[, NamedFormat])" enables the time stamp property to be output in plain text. The output is dependent on the current language setting. The language setting can be set using the VBS standard function SetLocale().

By implementing the second parameter of the FormatDate() function and further VBS standard functions such as Year, WeekDay, Day, Hour, Minute, Second enable the information, required by the user, to be split. Use the WeekdayName function to receive the name of the weekday for WeekDay.
Example:

'VBS87
Dim objTag
Dim lngCount
lngCount = 0
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
SetLocale("en-gb")
MsgBox FormatDateTime(objTag.TimeStamp)    'Output: e.g. 06/08/2002 9:07:50
MsgBox Year(objTag.TimeStamp)    'Output: e.g. 2002
MsgBox Month(objTag.TimeStamp)    'Output: e.g. 8
MsgBox Weekday(objTag.TimeStamp)    'Output: e.g. 3
MsgBox WeekdayName(Weekday(objTag.TimeStamp))    'Output: e.g. Tuesday
MsgBox Day(objTag.TimeStamp)    'Output: e.g. 6
MsgBox Hour(objTag.TimeStamp)    'Output: e.g. 9
MsgBox Minute(objTag.TimeStamp)    'Output: e.g. 7
MsgBox Second(objTag.TimeStamp)    'Output: e.g. 50
For lngCount = 0 To 4
    MsgBox FormatDateTime(objTag.TimeStamp, lngCount)
Next
'lngCount = 0: Output: e.g. 06/08/2002 9:07:50
'lngCount = 1: Output: e.g. 06 August 2002
'lngCount = 2: Output: e.g. 06/08/2002
'lngCount = 3: Output: e.g. 9:07:50
'lngCount = 4: Output: e.g. 9:07

Example:

The following example issues the time stamp of the tag "Tag1":

'VBS88
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.TimeStamp

See also

Tag Object (Page 137)
Alarms object (list) (Page 110)
TimeStepBase property

**Precision - TimeStepBase**

Defines the precision of the time stamp displayed in a table. Calculate the precision by multiplying the factor with the time unit. Enter factor "3" and time unit "1s" to display all values which were generated within 3 seconds in the same row, for example.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Exact</td>
<td>Only values with precisely the same time stamp are displayed in a table row.</td>
</tr>
<tr>
<td>100</td>
<td>100 ms</td>
<td>All values generated within 100 milliseconds are grouped in a table row.</td>
</tr>
<tr>
<td>250</td>
<td>250 ms</td>
<td>All values generated within 250 milliseconds are grouped in a table row.</td>
</tr>
<tr>
<td>500</td>
<td>500 ms</td>
<td>All values generated within 500 milliseconds are grouped in a table row.</td>
</tr>
<tr>
<td>1000</td>
<td>1 s</td>
<td>All values generated within 1 second are grouped in a table row.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TimeStepBase**. The data type is LONG.

TimeStepFactor property

**Precision - TimeStepFactor**

Defines the precision of the time stamp displayed in a table. Calculate the precision by multiplying the factor with the time unit. Enter factor "3" and time unit "1s" to display all values which were generated within 3 seconds in the same row. The factor entered is disabled if "Exact" is selected for the time unit or "0" is selected for "TimeStepBase".

The attribute can be assigned dynamic properties by means of the name **TimeStepFactor**. The data type is LONG.

TimeZone Property

**Description**

Defines the time zone used as a basis for displaying time values. Four settings are possible:

- Local time zone
- Server's time zone
- UTC (Universal Time Coordinated)
- Apply project settings (=> Use WinCC Explorer and access the computer's properties page to define the time mode specifically for the computer. The following are available for selection: WinCC V50 (Compatibility mode => Display as was standard in the individual display sections to V5 ), local time and UTC.)
See also

- WinCC Online Trend Control (before WinCC V7) (Page 295)
- WinCC Online Table Control (before WinCC V7) (Page 292)
- WinCC Function Trend Control (before WinCC V7) (Page 288)
- ScreenItem Object (Page 125)

**TitleColor property**

**Table header background - TitleColor**

Specifies the background color of the table headers. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name TitleColor. The data type is LONG.

**TitleCut property**

**Shorten contents - TitleCut**

Truncates the content of column headers if the column is insufficient.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The column headers are truncated.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The column headers are not truncated.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TitleCut. The data type is BOOLEAN.

**TitleCut property (before WinCC V7)**

**Description**

Defines whether the content of the fields of a title bar should be shortened if the column width is too small. Write/Read access.

**See also**

- WinCC Alarm Control (before WinCC V7) (Page 286)
- ScreenItem Object (Page 125)
TitleDarkShadowColor property

**Dark shading color - TitleDarkShadowColor**

Specifies the color of the dark side of shading. The button opens the "Color selection" dialog.

The setting is only active if the "Shading Color" option or "TitleStyle" is activated.

The attribute can be assigned dynamic properties by means of the name **TitleDarkShadowColor**. The data type is LONG.

TitleForeColor property

**Table header font color - TitleForeColor**

Specifies the color of the table header. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name **TitleForeColor**. The data type is LONG.

TitleGridLineColor property

**Color of the divider / header - TitleGridLineColor**

Defines the color of row/column dividers in the table header. The button opens the "Color selection" dialog.

The attribute can be assigned dynamic properties by means of the name **TitleGridLineColor**. The data type is LONG.

TitleLightShadowColor property

**Bright shading color - TitleLightShadowColor**

Specifies the color of the bright side of shading. The button opens the "Color selection" dialog.

The setting is only active if the "Shading Color" option or "TitleStyle" is activated.

The attribute can be assigned dynamic properties by means of the name **TitleLightShadowColor**. The data type is LONG.

Titeline Property

**Description**

TRUE, when the control has a title bar and it can be moved in Runtime. BOOLEAN write-read access.
See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

**TitleSort property**

**Sort by column title - TitleSort**

Defines how to trigger sorting by column title. You can only sort by column title if the "Auto-scrolling" option is deactivated.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>Sorting by column title is not possible.</td>
</tr>
<tr>
<td>1</td>
<td>With click</td>
<td>Sorting is triggered by clicking in the column header.</td>
</tr>
<tr>
<td>2</td>
<td>With double-click</td>
<td>Sorting is triggered by double-clicking in the column title.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TitleSort`. The data type is LONG.

**TitleStyle property**

**Shading color - TitleStyle**

Specifies whether to set a shading color for the table header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Flat</td>
<td>Disables the use of shading colors. Flat header style.</td>
</tr>
<tr>
<td>1</td>
<td>Button</td>
<td>Enables the use of shading colors. 3D representation of the header.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TitleStyle`. The data type is LONG.

**Toggle Property**

**Description**

TRUE, when the button or round button should lock after being operated in Runtime.

BOOLEAN write-read access.
See also

Round Button (Page 209)
ScreenItem Object (Page 125)

ToleranceHigh Property

Description

Defines or returns the limit value for "Tolerance high". The type of the evaluation (in percent or absolute) is defined in the "TypeToleranceHigh" property. The monitoring of the limit value is only valid if the "CheckToleranceHigh" property is set to "TRUE".

See also

Bar (Page 174)
ScreenItem Object (Page 125)

ToleranceLow Property

Description

Defines or returns the limit value for "Tolerance low". The type of the evaluation (in percent or absolute) is defined in the "TypeToleranceLow" property. The monitoring of the limit value is only valid if the "CheckToleranceLow" property is set to "TRUE".

See also

Bar (Page 174)
ScreenItem Object (Page 125)

Toolbar

Toolbar Property

Description

TRUE, when a toolbar is to be displayed. BOOLEAN write-read access.
See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

ToolbarAlignment property (before WinCC V7)

Description

Defines or returns the position of the toolbar. Write/Read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

ToolbarAlignment Property

Alignment - ToolbarAlignment

Defines the orientation of the Control toolbar.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Top</td>
<td>The toolbar is aligned to the top edge.</td>
</tr>
<tr>
<td>1</td>
<td>Bottom</td>
<td>The toolbar is aligned to the bottom edge.</td>
</tr>
<tr>
<td>2</td>
<td>Left</td>
<td>The toolbar is aligned to the left edge.</td>
</tr>
<tr>
<td>3</td>
<td>Right</td>
<td>The toolbar is aligned to the right edge.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarAlignment. The data type is LONG.

ToolbarBackColor property

BackgroundColor - ToolbarBackColor

Specifies the background color of the toolbar. Open the "Color selection" dialog by clicking the button.
The background color you configured is only displayed if the "Display" option is activated or "ToolbarUseBackColor" is "TRUE".

The attribute can be assigned dynamic properties by means of the name ToolbarBackColor. The data type is LONG.

### ToolbarButtonActive property

**Active - ToolbarButtonActive**

Activates a button function in Runtime. Clicking the button in Runtime triggers the corresponding function.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The button function is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The button function is disabled. You can assign custom functions to the button by means of scripting.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarButtonActive. The data type is BOOLEAN.

### ToolbarButtonAdd property

**New - ToolbarButtonAdd**

Creates a new, user-defined button function. The name set by WinCC can be edited in the "Object name" field.

The attribute can be assigned dynamic properties by means of the name ToolbarButtonAdd. The data type is STRING.

### ToolbarButtonBeginGroup property

**Separator - ToolbarButtonBeginGroup**

Inserts a leading separator (vertical line) for the selected button function on the toolbar. These separators can be used to group the icons of the button functions.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>A separator prefix is inserted for the button function selected.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A separator prefix is not inserted for the button function selected.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarButtonBeginGroup. The data type is BOOLEAN.
ToolbarButtonClick AlarmControl property

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>21</td>
<td>&quot;Next message&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>22</td>
<td>&quot;Last message&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Message list&quot;</td>
<td>23</td>
<td>&quot;Info text dialog&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Short-term archive list&quot;</td>
<td>24</td>
<td>&quot;Comments dialog&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Long-term archive list&quot;</td>
<td>25</td>
<td>&quot;Loop In Alarm&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Lock List&quot;</td>
<td>26</td>
<td>&quot;Lock message&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Hit List&quot;</td>
<td>27</td>
<td>&quot;Enable message&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;List of messages to be hidden&quot;</td>
<td>28</td>
<td>&quot;Hide messages&quot;</td>
</tr>
<tr>
<td>9</td>
<td>&quot;Ackn. Central Signaling Devices&quot;</td>
<td>29</td>
<td>&quot;Unhide messages&quot;</td>
</tr>
<tr>
<td>10</td>
<td>&quot;Single acknowledgment&quot;</td>
<td>30</td>
<td>&quot;Sort dialog&quot;</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Group acknowledgement&quot;</td>
<td>31</td>
<td>&quot;Time base dialog&quot;</td>
</tr>
<tr>
<td>12</td>
<td>&quot;Autoscroll&quot;</td>
<td>32</td>
<td>&quot;Copy rows&quot;</td>
</tr>
<tr>
<td>13</td>
<td>&quot;Selection dialog&quot;</td>
<td>33</td>
<td>&quot;Connect backup&quot;</td>
</tr>
<tr>
<td>14</td>
<td>&quot;Display options dialog&quot;</td>
<td>34</td>
<td>&quot;Disconnect backup&quot;</td>
</tr>
<tr>
<td>15</td>
<td>&quot;Lock dialog&quot;</td>
<td>36</td>
<td>&quot;First page&quot;</td>
</tr>
<tr>
<td>16</td>
<td>&quot;Print&quot;</td>
<td>37</td>
<td>&quot;Previous page&quot;</td>
</tr>
<tr>
<td>17</td>
<td>&quot;Export data&quot;</td>
<td>38</td>
<td>&quot;Next page&quot;</td>
</tr>
<tr>
<td>18</td>
<td>&quot;Autoscroll&quot;</td>
<td>39</td>
<td>&quot;Last page&quot;</td>
</tr>
<tr>
<td>19</td>
<td>&quot;First message&quot;</td>
<td>1001</td>
<td>&quot;User-defined 1&quot;</td>
</tr>
<tr>
<td>20</td>
<td>&quot;Previous message&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarButtonClick. The data type is LONG.

ToolbarButtonClick FunctionTrendControl property

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>13</td>
<td>&quot;Select time range&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>14</td>
<td>&quot;Previous trend&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Zoom area&quot;</td>
<td>15</td>
<td>&quot;Next trend&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Zoom +/-&quot;</td>
<td>16</td>
<td>&quot;Stop&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Zoom X axis +/-&quot;</td>
<td></td>
<td>&quot;Start&quot;</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name ToolbarButtonClick. The data type is LONG.

**ToolbarButtonClick OnlineTableControl property**

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>13</td>
<td>&quot;Next column&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>14</td>
<td>&quot;Stop&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;First data record&quot;</td>
<td>14</td>
<td>&quot;Start&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Previous data record&quot;</td>
<td>15</td>
<td>&quot;Print&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Next data record&quot;</td>
<td>20</td>
<td>&quot;Export data&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Last data record&quot;</td>
<td>16</td>
<td>&quot;Define statistics area&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Edit&quot;</td>
<td>17</td>
<td>&quot;Calculate statistics&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;Copy rows&quot;</td>
<td>18</td>
<td>&quot;Connect backup&quot;</td>
</tr>
<tr>
<td>9</td>
<td>&quot;Select data connection&quot;</td>
<td>19</td>
<td>&quot;Disconnect backup&quot;</td>
</tr>
<tr>
<td>10</td>
<td>&quot;Select columns&quot;</td>
<td>21</td>
<td>&quot;Create archive value&quot;</td>
</tr>
<tr>
<td>11</td>
<td>&quot;Select time range&quot;</td>
<td>1001</td>
<td>&quot;User-defined 1&quot;</td>
</tr>
<tr>
<td>12</td>
<td>&quot;Previous column&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarButtonClick. The data type is LONG.

**ToolbarButtonClick OnlineTrendControl property**

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>17</td>
<td>&quot;Select time range&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>18</td>
<td>&quot;Previous trend&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;First data record&quot;</td>
<td>19</td>
<td>&quot;Next trend&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Previous data record&quot;</td>
<td>20</td>
<td>&quot;Stop&quot;</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ToolbarButtonClick. The data type is LONG.
The attribute can be assigned dynamic properties by means of the name `ToolbarButtonClick`. The data type is LONG.

### ToolbarButtonClick RulerControl property

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Ruler window&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Statistics range&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Statistics&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Print&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Export data&quot;</td>
</tr>
<tr>
<td>1001</td>
<td>&quot;User-defined 1&quot;</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolbarButtonClick`. The data type is LONG.

### ToolbarButtonClick SysdiagControl property

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Configuration dialog&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Overview&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Back&quot;</td>
</tr>
</tbody>
</table>
4  "Open"
5  "Diagnostics buffer"
6  "Update diagnostic buffer"
7  "Sort dialog"
8  "Print"
9  "Export data"

The attribute can be assigned dynamic properties using the name `ToolbarButtonClick`. The data type is LONG.

**ToolbarButtonClick UserAdminControl property**

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>9</td>
<td>&quot;Log on&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>10</td>
<td>&quot;Log off&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Group list&quot;</td>
<td>11</td>
<td>&quot;Change password&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;User list&quot;</td>
<td>12</td>
<td>&quot;Import user administration&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Edit&quot;</td>
<td>13</td>
<td>&quot;Export user administration&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Add&quot;</td>
<td>14</td>
<td>&quot;Sort dialog&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Authorization levels&quot;</td>
<td>15</td>
<td>&quot;Print&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;Delete&quot;</td>
<td>16</td>
<td>&quot;Export data&quot;</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name `ToolbarButtonClick`. The data type is LONG.

**ToolbarButtonDown UserArchiveControl property**

**ToolbarButtonClick**

Triggers the function linked to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

<table>
<thead>
<tr>
<th>ID</th>
<th>Button function</th>
<th>ID</th>
<th>Button function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;Help&quot;</td>
<td>12</td>
<td>&quot;Read tags&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Configuration dialog&quot;</td>
<td>13</td>
<td>&quot;Write tags&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Select data connection&quot;</td>
<td>14</td>
<td>&quot;Import archive&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;First row&quot;</td>
<td>15</td>
<td>&quot;Export archive&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Previous row&quot;</td>
<td>16</td>
<td>&quot;Sort dialog&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Next row&quot;</td>
<td>17</td>
<td>&quot;Selection dialog&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Last row&quot;</td>
<td>18</td>
<td>&quot;Print&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;Delete rows&quot;</td>
<td>20</td>
<td>&quot;Export data&quot;</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `ToolbarButtonClick`. The data type is LONG.

**ToolbarButtonCount property**

**ToolbarButtonCount**

Defines the number of configurable button functions.

The attribute can be assigned dynamic properties by means of the name `ToolbarButtonCount`. The data type is LONG.

**ToolbarButtonEnabled property**

**ToolbarButtonEnabled**

Enables operation of custom toolbar buttons.

The attribute can be assigned dynamic properties by means of the name `ToolbarButtonEnabled`. The data type is BOOLEAN.

**ToolbarButtonHotKey property**

**Hotkey - ToolbarButtonHotKey**

Shows the hotkey for a button function selected.

You create or edit a hotkey by clicking in the "Hotkey" field and pressing the button or key shortcut required.

The attribute can be assigned dynamic properties by means of the name `ToolbarButtonHotKey`. The data type is LONG.

**ToolbarButtonID property**

**Object ID - ToolbarButtonID**

Unique ID number for the selected button function. WinCC assigns this read only ID number.

The attribute can be assigned dynamic properties by means of the name `ToolbarButtonID`. The data type is LONG.
ToolbarButtonIndex property

ToolbarButtonIndex
References a button function. Using this attribute you can assign the values of other attributes to a specific button function.

Values between 0 and "ToolbarButtonCount" minus 1 are valid for "ToolbarButtonCount". Attribute "ToolbarButtonCount" defines the number of configurable button functions.

The "ToolbarButtonIndex" attribute can be assigned dynamic properties by means of attribute ToolbarButtonRepos. The data type is LONG.

ToolbarButtonLocked property

ToolbarButtonLocked
Enables/disables the display of the pressed state of a user-defined toolbar button.

The attribute can be assigned dynamic properties by means of the name ToolbarButtonLocked. The data type is BOOLEAN.

ToolbarButtonName property

Object name - ToolbarButtonName
Shows the name for the selected button function. You rename user-defined button functions.

The "ToolbarButtonName" attribute can be assigned dynamic properties by means of attribute ToolbarButtonRename. The data type is STRING.

ToolbarButtonPasswordLevel property

Operator authorization - ToolbarButtonPasswordLevel
Shows the authorization for a button function selected. You can edit the authorization using the selection button.

Authorizations are configured in the "User Administrator" editor.

The attribute can be assigned dynamic properties by means of the name ToolbarButtonPasswordLevel. The data type is LONG.

ToolbarButtonRemove property

Remove - ToolbarButtonRemove
Removes the selected button function from the list. Only user-defined button functions can be removed.
The attribute can be assigned dynamic properties by means of the name **ToolbarButtonRemove**. The data type is STRING.

**ToolbarButtonRename property**

**ToolbarButtonRename**

Renames a custom toolbar element which is referenced by means of "ToolbarButtonIndex" attribute.

The attribute for custom elements can be assigned dynamic properties by means of the name **ToolbarButtonRename**. "ToolbarButtonRename" also sets a dynamic attribute "ToolbarButtonName". The data type is STRING.

**ToolbarButtonRepos property**

**Up/Down - ToolbarButtonRepos**

Changes the sorting order of button functions. "Up" and "Down" move the button function selected up or down in the list. This moves the button function in the toolbar of a Control towards the front or towards the back.

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonRepos**. The data type is LONG.

**ToolbarButtonSize property**

**Button size in pixels - ToolbarButtonSize**

Defines the size of the buttons in the toolbar.

With the maximum value of 280 pixels, the button size can be ten times larger than the original size of 28 pixels.

The following behavior results for the button size depending on the configured value:

<table>
<thead>
<tr>
<th>Value of the button size</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value &lt; 0</td>
<td>Invalid value. The most recent valid value is used.</td>
</tr>
<tr>
<td>0 ≤ value ≤ original size of button</td>
<td>The original size of the button is used. &quot;ToolbarButtonSize&quot; is set to the default (= 0).</td>
</tr>
<tr>
<td>Original size of the button &lt; value ≤ maximum value</td>
<td>The configured value is used.</td>
</tr>
<tr>
<td>Maximum value &lt; value</td>
<td>Invalid value. The most recent valid value is used.</td>
</tr>
</tbody>
</table>

With a large button size, please note that in some cases not all buttons may be displayed in the control. To show all activated buttons in Runtime, you must therefore extend the control or activate fewer buttons as required.

You can create a dedicated button to zoom in or out of the toolbar in Runtime. For instructions, refer to the example "How to configure a user-defined toolbar button with a self-created selection dialog" in the documentation "VBS for Creating Procedures and Actions".
The attribute can be assigned dynamic properties by means of the name `ToolBarButtonSize`. The data type is LONG.

**ToolBarButtonTooltipText property**

**ToolBarButtonTooltipText**

Specifies the tooltip text for the button.

The attribute can be assigned dynamic properties by means of the name `ToolBarButtonTooltipText`. The data type is STRING.

**ToolBarButtonUserDefined property**

**ToolBarButtonUserDefined**

Indicates whether the project engineer has added a new user-defined toolbar button.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The toolbar button is assigned a user-defined function.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The toolbar button is defined by the system.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolBarButtonUserDefined`. The data type is BOOLEAN.

**ToolBarButtonVisible property**

**Button functions - ToolBarButtonVisible**

Select the button functions to be displayed in the toolbar from the list.

Click a list entry to adapt the properties, or to change the position in the status bar of the Control by means of the "Up" and "Down" buttons.

The attribute can be assigned dynamic properties by means of the name `ToolBarButtonVisible`. The data type is BOOLEAN.

**ToolBarButtons Property**

**Description**

Defines or returns the buttons contained in the toolbar by setting or resetting the corresponding bits. Each button is assigned a bit. There are no limitations as to the bit combinations.

Bit - Value (hex) ; Value (dec) ; Button:

- 0 - 0x00000001; 1; Message List
- 1 - 0x00000002; 2; Short-term archive list
2 - 0x00000004; 4; Long-term archive list
3 - 0x00000008; 8; Acknowledgment of central signaling device
4 - 0x00000010; 16; Single Acknowledgment
5 - 0x00000020; 32; Group acknowledgment
6 - 0x00000040; 64; Autoscroll
7 - 0x00000080; 128; Selection Dialog
8 - 0x00000100; 256; Lock Dialog
9 - 0x00000200; 512; Print message log
11 - 0x00000800; 2048; Emergency acknowledgment
12 - 0x00001000; 4096; First message
13 - 0x00002000; 8192; Last message
14 - 0x00004000; 16384; Next message
15 - 0x00008000; 32768; Previous message
16 - 0x00010000; 65536; Infotext Dialog
17 - 0x00020000; 131072; Comment Dialog
18 - 0x00040000; 262144; Loop in Alarm
20 - 0x00100000; 1048576; Print current view
21 - 0x00200000; 2097152; Lock list
22 - 0x00400000; 4194304; Lock/release message
23 - 0x00800000; 8388608; Sorting Dialog
24 - 0x01000000; 16777216; Time basis dialog
25 - 0x02000000; 33554432; Hit list

In order to display more buttons, their values must be logically linked with OR. Write/Read access.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)
**ToolbarHotKeys Property**

**Description**

Defines or returns hotkeys of the buttons in the toolbar. Write/Read access.

**See also**

WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

**ToolbarShowTooltips property**

**Tooltips - ToolbarShowTooltips**

Enables the display of tooltips for the button functions in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of tooltips.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of tooltips.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolbarShowTooltips`. The data type is BOOLEAN.

Attribute "ToolbarButtonTooltipText" defines the tooltip text.

**ToolbarUseBackColor property**

**Show background color - ToolbarUseBackColor**

Enables the display of the background color for a toolbar.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of the background color of a toolbar.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of the background color of a toolbar.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolbarUseBackColor`. The data type is BOOLEAN.
ToolBarUseHotKeys property

Hotkeys - ToolBarUseHotKeys
Activates the hotkeys for button functions in Runtime. Insert the hotkeys for button functions in the "Hotkey" field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The hotkeys are activated.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The hotkeys are deactivated.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolBarUseHotKeys`. The data type is BOOLEAN.

ToolBarVisible property

Show toolbar - ToolBarVisible
Enables the display of the Control toolbar.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of the toolbar.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of the toolbar.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ToolBarVisible`. The data type is BOOLEAN.

ToolTip - TrendLower

ToolTipText Property

Description
Defines or returns the text to be displayed as a tooltip when the mouse is positioned over the object.

STRING (write-read access)
Example:

The following example assigns a tool tip text to every object in the picture "NewPDL1": The picture "NewPDL1" comprises only objects containing the ToolTipText property:

'VBS89
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
'Assign tooltiptexts to the objects
objScrItem.ToolTipText = "Name of object is " & strName
Next

See also

Radio box (Page 207)
Status display (Page 199)
Connector (Page 167)
Text list (Page 196)
Static text (Page 165)
Slider (Page 212)
Group Display (Page 194)
Rounded rectangle (Page 162)
Round Button (Page 209)
Rectangle (Page 159)
Polyline (Page 158)
Polygon (Page 156)
OLE object (Page 191)
Line (Page 154)
Pie segment (Page 152)
Circular arc (Page 151)
Circle (Page 149)
Group (Page 300)
Graphic Object (Page 187)
Top Property

Function

Defines or returns the Y-coordinate of an object (measured from the top left edge of the picture) in pixels. The Y-coordinate relates to the top left corner of the rectangle enclosing the object.

LONG (write-read access)

Example:

The following example shifts all objects in the picture "NewPDL1" 5 pixels upwards:

```
'VBS90
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
objScrItem.Top = objScrItem.Top - 5
Next
```

See also

Left Property (Page 476)

ScreenItem Object (Page 125)
TopConnectedConnectionPointIndex Property

Description
Specifies or sets the index number of the top connecting point.
LONG write-read access.

See also
Connector (Page 167)
ScreenItem Object (Page 125)

TopConnectedObjectName Property

Description
Specifies or sets the object name of the object which is docked on at the bottom connecting point.
LONG write-read access.

See also
Connector (Page 167)
ScreenItem Object (Page 125)

Transparency property

Description
Defines and returns the percentage transparency of the object.
0 = no transparency; 100 = complete transparency (invisible)
The text and fields of the graphic objects are only transparent at "100."
In runtime, a completely transparent object (invisible) is also functional.

Transparent Property

Description
TRUE, when the button appears completely filled in the color specified in “BackColor”.
BOOLEAN write-read access.
See also

WinCC Push Button Control (Page 269)
ScreenItem Object (Page 125)

Trend Property

Description

TRUE, when the tendency (rising or falling) of the measuring value being monitored should be displayed by a small arrow. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

TrendActualize property

Update -TrendActualize

Enables the update of a selected trend.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables updates of the trend selected.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables updates of the trend selected. This setting can be useful when comparing a logged trend with a current trend.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendActualize. The data type is BOOLEAN.

TrendAdd property

New - TrendAdd

Creates a new trend.

The attribute can be assigned dynamic properties by means of the name TrendAdd. The data type is STRING.

TrendAutoRangeBeginTagName property

This attribute sets the low limit tag for the range of values if the range of values is calculated automatically by means of online tags.
The attribute can be assigned dynamic properties by means of the name `TrendAutoRangeBeginTagName`. The data type is STRING.

**TrendAutoRangeBeginValue property**

**TrendAutoRangeBeginValue**
This attribute sets the low limit tag for the range of values if the range of values is calculated based on the configuration of high and low limits.

The attribute can be assigned dynamic properties by means of the name `TrendAutoRangeBeginValue`. The data type is DOUBLE.

**TrendAutoRangeEndTagName property**

**TrendAutoRangeEndTagName**
This attribute sets the high limit tag for the range of values if the range of values is calculated automatically by means of online tags.

The attribute can be assigned dynamic properties by means of the name `TrendAutoRangeEndTagName`. The data type is STRING.

**TrendAutoRangeEndValue property**

**TrendAutoRangeEndValue**
This attribute sets the high limit tag for the range of values if the range of values is calculated based on the configuration of high and low limits.

The attribute can be assigned dynamic properties by means of the name `TrendAutoRangeEndValue`. The data type is DOUBLE.

**TrendAutoRangeSource property**

**TrendAutoRangeSource**
Defines the mode for automatic calculation of the range of values of trend data.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Display data</td>
<td>The range of values is calculated automatically based on the data displayed.</td>
</tr>
<tr>
<td>1</td>
<td>Value range</td>
<td>The range of values is defined based on its configured low and high limit. The low and high limits are emulated in the &quot;TrendAutoRangeBeginValue&quot; and &quot;TrendAutoRangeEndValue&quot; attributes.</td>
</tr>
<tr>
<td>2</td>
<td>Online tags</td>
<td>The low and high limits of the range of values are derived from the values of connected online tags. The low and high limits are emulated in the &quot;TrendAutoRangeBeginTagName&quot; and &quot;TrendAutoRangeEndTagName&quot; attributes.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `TrendAutoRangeSource`. The data type is LONG.

**TrendBeginTime property**

**Start time - TrendBeginTime**

Defines the start time of the time range for data transfer to the selected trend.

The attribute can be assigned dynamic properties by means of the name `TrendBeginTime`. The data type is Date.

**TrendColor property**

**Trend color - TrendColor**

Specifies the trend color. Open the "Color selection" dialog by clicking the button.

The attribute can be assigned dynamic properties by means of the name `TrendColor`. The data type is LONG.

**LTrendColor property (before WinCC V7)**

**Description**

Determines the color of the trend display or returns it.

The trend display indicates the tendency (rising or falling) of the measuring value being monitored by a small arrow. In order to activate the trend display, the Trend property must be set to "True". LONG write-read access.

**See also**

Bar (Page 174)

ScreenItem Object (Page 125)

**TrendCount property**

**TrendCount**

Defines the number of configured trends.

The attribute can be assigned dynamic properties by means of the name `TrendCount`. The data type is LONG.
TrendEndTime property

End time - TrendEndTime

Defines the end of the time range for data connections of a selected trend.

The attribute can be assigned dynamic properties by means of the name TrendEndTime. The data type is Date.

TrendExtendedColorSet property

Extended - TrendExtendedColorSet

Enables configuration of the point and fill colors and the display of colors in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The &quot;Point color&quot; and &quot;Fill color&quot; field settings can be configured and are active in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The &quot;Point color&quot; and &quot;Fill color&quot; field settings cannot be configured and are inactive in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendExtendedColorSet. The data type is BOOLEAN.

TrendFill property

Filled - TrendFill

Specifies if the area beneath the trend is to be filled.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The area beneath the trend is shown filled. You can define the trend color as fill color if the &quot;Advanced&quot; option is deactivated. The text background is displayed in the trend color for the trend type &quot;Values&quot;. The background color of the control is used as text color.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The trend is not visualized with fill color.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendFill. The data type is BOOLEAN.

TrendFillColor property

Fill color - TrendFillColor

Specifies the fill color of the trend. The text fill color is specified for the trend type "Values".

The fill color is used if the "Filled" option is activated or "TrendFill" is "TRUE". Open the "Color selection" dialog by clicking the button.
The configuration is only possible if the "Advanced" option is activated or "TrendExtendedColorSet" is "TRUE".

The attribute can be assigned dynamic properties by means of the name `TrendFillColor`. The data type is LONG.

**TrendIndex property**

**TrendIndex**

References a configured trend. Using this attribute you can assign the values of other attributes to a specific trend. The index must always be set before you change the properties of a trend in runtime.

Values between 0 and "TrendIndex" minus 1 are valid for "TrendCount". Attribute "TrendCount" defines the number of trends configured.

The "TrendIndex" attribute can be assigned dynamic properties by means of attribute `TrendRepos`. The data type is LONG.

**TrendLabel property**

**Label - TrendLabel**

Defines the label of the trend selected. The label is displayed in Runtime if the value at attribute "UseTrendNameAsLabel" is "FALSE".

The attribute can be assigned dynamic properties by means of the name `TrendLabel`. The data type is STRING.

**TrendLineStyle property**

**Line style - TrendLineStyle**

Defines the line style for trend visualization.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Solid</td>
<td>The trend is visualized as solid line.</td>
</tr>
<tr>
<td>1</td>
<td>Dashed</td>
<td>The trend is visualized as dashed line.</td>
</tr>
<tr>
<td>2</td>
<td>Dotted</td>
<td>The trend is visualized as dotted line.</td>
</tr>
<tr>
<td>3</td>
<td>Dash dot</td>
<td>The trend is visualized as dot-dash line.</td>
</tr>
<tr>
<td>4</td>
<td>Dash Dot Dot</td>
<td>The trend is visualized as dash-dot-dot line.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendLineStyle`. The data type is LONG.
TrendLineType property

Trend type - TrendLineType

Defines how to visualize a trend.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>Only the dots are displayed.</td>
</tr>
<tr>
<td>1</td>
<td>Connect dots linearly</td>
<td>Visualizes a trend with linear interconnection of points.</td>
</tr>
<tr>
<td>2</td>
<td>Stepped</td>
<td>Visualizes a stepped trend and its interconnected points.</td>
</tr>
<tr>
<td>3</td>
<td>Values</td>
<td>Can only be configured with OnlineTrendControl. A value is displayed at each time stamp or at the main grid line of the time axis instead of trend points.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendLineType`. The data type is LONG.

TrendLineWidth property

Line weight - TrendLineWidth

Defines the line weight of the line displayed.

The attribute can be assigned dynamic properties by means of the name `TrendLineWidth`. The data type is LONG.

TrendLowerLimit property

TrendLowerLimit

Specifies the low limit of a tag. The values are identified based on the color set in "TrendLowerLimitColor" if the tag value is less than "TrendLowerLimit". This setting is only active if the value at attribute "TrendLowerLimitColoring" is "TRUE".

The attribute can be assigned dynamic properties by means of the name `TrendLowerLimit`. The data type is DOUBLE.

TrendLowerLimitColor property

TrendLowerLimitColor

Specifies the color of tag values which are less than the value at "TrendLowerLimit". This setting is only active if the value at attribute "TrendLowerLimitColoring" is "TRUE".

The attribute can be assigned dynamic properties by means of the name `TrendLowerLimitColor`. The data type is LONG.
TrendLowerLimitColoring property

TrendLowerLimitColoring Enables the "TrendLowerLimitColor" attribute for identifying tag values which are less than the value at "TrendLowerLimitValue".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Attribute &quot;TrendLowerLimitColor&quot; is active.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Attribute &quot;TrendLowerLimitColor&quot; is inactive.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendLowerLimitColoring. The data type is BOOLEAN.

TrendMeasure - TrendVisible

TrendMeasurePoints property

Number of measurement points - TrendMeasurePoints Defines the number of measurement points for visualization of selected trends.

Defines the number of value pairs provided to the trend from a user archive.

The attribute can be assigned dynamic properties by means of the name TrendMeasurePoints. The data type is LONG.

TrendName property

Object name - TrendName Displays the name of the selected trend. The name is defined on the "Trends" tab.

The "TrendName" attribute can be assigned dynamic properties by means of attribute TrendRename. The data type is STRING.

TrendPointColor property

Point color - TrendPointColor Defines the color of trend points. Open the "Color selection" dialog by clicking the button.

The configuration is only possible if the "Advanced" option is activated or "TrendExtendedColorSet" is "TRUE".

The attribute can be assigned dynamic properties by means of the name TrendPointColor. The data type is LONG.
**TrendPointStyle property**

**Dot type - TrendPointStyle**

Defines the dot style for trend visualization.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>The points are not displayed.</td>
</tr>
<tr>
<td>1</td>
<td>Dots</td>
<td>The trend points are visualized with a size of one pixel. The setting in the &quot;Dot width&quot; field is deactivated.</td>
</tr>
<tr>
<td>2</td>
<td>Squares</td>
<td>The dots are displayed as squares. The setting in the &quot;Dot width&quot; field is active.</td>
</tr>
<tr>
<td>3</td>
<td>Circles</td>
<td>The dots are displayed as circles. The setting in the &quot;Dot width&quot; field is active.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TrendPointStyle**. The data type is LONG.

**TrendPointWidth property**

**Dot width - TrendPointWidth**

Sets the dot width in pixels. You can only define the dot width for the "square" and "circular" type.

The attribute can be assigned dynamic properties by means of the name **TrendPointWidth**. The data type is LONG.

**TrendProvider property**

**Data source - TrendProvider**

Specifies the data source for a selected trend.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>No data source configured for implementation in Runtime by means of script.</td>
</tr>
<tr>
<td>1</td>
<td>Archive tags</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
<tr>
<td>2</td>
<td>Online tags</td>
<td>Data source with online tags derived from tag management.</td>
</tr>
<tr>
<td>3</td>
<td>User archive</td>
<td>Data source with columns of a user archive.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TrendProvider**. The data type is LONG.
TrendProviderCLSID_FunctionTrend property

TrendProviderCLSID_FunctionTrend
Indicates the data source of the trend selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>{416A09D2-8B5A-11D2-8B81-006097A45D48}</td>
<td>No data source configured for implementation in Runtime by means of script.</td>
</tr>
<tr>
<td>{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
<tr>
<td>{2DC9B1C8-4FC1-41B1-B354-3E469A13FBFD}</td>
<td>Data source with online tags derived from tag management.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendProviderCLSID. The data type is STRING.

TrendProviderCLSID_OnlineTrend property

TrendProviderCLSID_OnlineTrend
Indicates the data source of the trend selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>{416A09D2-8B5A-11D2-8B81-006097A45D48}</td>
<td>No data source configured for implementation in Runtime by means of script.</td>
</tr>
<tr>
<td>{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
<tr>
<td>{2DC9B1C8-4FC1-41B1-B354-3E469A13FBFD}</td>
<td>Data source with online tags derived from tag management.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendProviderCLSID. The data type is STRING.

TrendRangeType property

Time range setting - TrendRangeType
Defines the time range for providing data to the selected trend.
You can only define the number of measuring points if you select user archives as the data source.
The following configuration options are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Time range</td>
<td>Defines the start time and the time range for the data connection.</td>
</tr>
<tr>
<td>1</td>
<td>Start to end time</td>
<td>Defines the start and end time for the data connection.</td>
</tr>
<tr>
<td>2</td>
<td>Number of measurement points</td>
<td>Defines the start time and the number of measurement points for the data connection.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendRangeType`. The data type is LONG.

**TrendRemove property**

Remove - TrendRemove

Removes selected trends from the list.

The attribute can be assigned dynamic properties by means of the name `TrendRemove`. The data type is STRING.

**TrendRename property**

TrendRename

Renames a trend which is referenced by means of "TrendIndex" attribute.

The attribute can be assigned dynamic properties by means of the name `TrendRename`. "TrendRename" also sets a dynamic attribute "TrendName". The data type is STRING.

**TrendRepos property**

Up/Down - TrendRepos

Repositions the trend in the trend window. "Up" and "Down" move the selected trend up or down in the list. This moves the trend towards the foreground or background for visualization in Runtime.

The attribute can be assigned dynamic properties by means of the name `TrendRepos`. The data type is LONG.

**TrendSelectTagName property**

TrendSelectTagName

Opens a dialog for selecting the tag name for the source of Y axis data in WinCC OnlineTrendControl. Programmers can set this attribute to allow users to select a tag name by means of a button, for example.
The attribute can be assigned dynamic properties by means of the name `TrendSelectTagName`. The data type is BOOLEAN.

**TrendSelectTagNameX property**

`TrendSelectTagNameX`  
Opens a dialog for selecting the tag name for the source of X axis data in WinCC FunctionTrendControl. Programmers can set this attribute to allow users to select a tag name by means of a button, for example.

The attribute can be assigned dynamic properties by means of the name `TrendSelectTagNameX`. The data type is BOOLEAN.

**TrendSelectTagNameY property**

`TrendSelectTagNameY`  
Opens a dialog for selecting the tag name for the source of Y axis data in WinCC FunctionTrendControl. Programmers can set this attribute to allow users to select a tag name by means of a button, for example.

The attribute can be assigned dynamic properties by means of the name `TrendSelectTagNameY`. The data type is BOOLEAN.

**TrendShowAlarms property**

**Displaying alarms - TrendShowAlarms**

If you have connected online tags and activate the "Show alarms" option, you can have the assigned message displayed as a symbol and tooltip for the trend values with limit violation. Limit monitoring must be configured in the alarm logging for the online tag for this.

The red symbol indicates a high or low limit violation. The tooltip contains the message text and information text of the message. If you have configured "Loop in Alarm" with the "OpenPicture" function for the message, you can jump to the assigned picture by double-clicking this symbol.

The attribute can be assigned dynamic properties by means of the name `TrendShowAlarms`. The data type is BOOLEAN.

**TrendState property**

**TrendState**  
Shows the status of the data link of the selected curve in Runtime.

The attribute can be made dynamic with the name `TrendState`. The data type is LONG.
TrendTagName property

**Tag name - TrendTagName**

Displays the name of connected tags. Use the Open button to open a dialog for selecting an online or archive tag.

The attribute can be assigned dynamic properties by means of the name `TrendTagName`. The data type is STRING.

TrendTagNameX property

**Tag Name X / Column X - TrendTagNameX**

Shows the name of interconnected tags or of the column for the X axis. Using the selection button, select a tag or a column for the data source you configured.

The attribute can be assigned dynamic properties by means of the name `TrendTagNameX`. The data type is STRING.

TrendTagNameY property

**Tag Name Y / Column Y - TrendTagNameY**

Shows the name of interconnected tags or of the column for the Y axis. Using the selection button, select a tag or a column for the data source you configured.

The attribute can be assigned dynamic properties by means of the name `TrendTagNameY`. The data type is STRING.

TrendTimeAxis property

**Time axis - TrendTimeAxis**

Defines the time axis to be used for the trend selected. Define the available time axes in the "Time axes" tab.

The attribute can be assigned dynamic properties by means of the name `TrendTimeAxis`. The data type is STRING.

TrendTimeRangeBase property

**Time Range - TrendTimeRangeBase**

Defines the time unit for calculating the time range.
The following time units are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500 ms</td>
</tr>
<tr>
<td>1000</td>
<td>1 second</td>
</tr>
<tr>
<td>60000</td>
<td>1 minute</td>
</tr>
<tr>
<td>3600000</td>
<td>1 hour</td>
</tr>
<tr>
<td>86400000</td>
<td>1 day</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendTimeRangeBase`. The data type is LONG.

**TrendTimeRangeFactor property**

**Time range - TrendTimeRangeFactor**

Defines the factor for calculating the time range. Only integer factors are valid.

The attribute can be assigned dynamic properties by means of the name `TrendTimeRangeFactor`. The data type is SHORT.

**TrendTrendWindow property**

**Trend window - TrendTrendWindow**

Defines the trend window for visualizing the trend selected. Define the available trend windows in the "Trend window" tab.

The attribute can be assigned dynamic properties by means of the name `TrendTrendWindow`. The data type is STRING.

**TrendUncertainColor property**

**TrendUncertainColor**

Value are in uncertain state if the initial value is unknown after runtime has been activated, or if a substitute value is used. Set attribute "TrendUncertainColor" to define the color identifier of these values. The "TrendUncertainColoring" attribute determines whether or not this setting is evaluated.

The attribute can be assigned dynamic properties by means of the name `TrendUncertainColor`. The data type is LONG.
**TrendUncertainColoring property**

**TrendUncertainColoring**

Value are in uncertain state if the initial value is unknown after runtime has been activated, or if a substitute value is used. The "TrendUncertainColoring" attribute is used to enable identification of such values based on the color set in "TrendUncertainColor".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The settings of the &quot;TrendUncertainColor&quot; attribute are active.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The settings of the &quot;TrendUncertainColor&quot; attribute are inactive.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TrendUncertainColoring**. The data type is BOOLEAN.

**TrendUpperLimit property**

**TrendUpperLimit**

Specifies the high limit of a tag. The values are identified based on the color set in "TrendUpperLimitColor" if the tag value exceeds the "TrendUpperLimit". This setting is only active if the value at attribute "TrendUpperLimitColoring" is "TRUE".

The attribute can be assigned dynamic properties by means of the name **TrendUpperLimit**. The data type is DOUBLE.

**TrendUpperLimitColor property**

**TrendUpperLimitColor**

Specifies the color of tag values which are less than the value at "TrendLowerLimit". This setting is only active if the value at attribute "TrendUpperLimitColoring" is "TRUE".

The attribute can be assigned dynamic properties by means of the name **TrendUpperLimitColor**. The data type is LONG.

**TrendUpperLimitColoring property**

**TrendUpperLimitColoring**

Enables the "TrendUpperLimitColor" attribute for identifying tag values which are less than the value at "TrendUpperLimit".

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The setting of the &quot;TrendUpperLimitColor&quot; attribute is active.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The setting of the &quot;TrendUpperLimitColor&quot; attribute is inactive.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name **TrendUpperLimitColoring**. The data type is BOOLEAN.

### TrendValueAlignment property

**Alignment - TrendValueAlignment**

Specifies the alignment of the displayed values for the trend type "Values".

The following settings are available depending on the writing direction of the trend:

- The writing direction of the trend is "from right" or "from left"

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bottom</td>
<td>The values are displayed at the bottom in the trend window.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The values are displayed centered in the trend window.</td>
</tr>
<tr>
<td>2</td>
<td>Top</td>
<td>The values are displayed at the top in the trend window.</td>
</tr>
</tbody>
</table>

- The writing direction of the trend is "from top" or "from bottom"

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Left</td>
<td>The values are displayed on the left in the trend window.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The values are displayed centered in the trend window.</td>
</tr>
<tr>
<td>2</td>
<td>Right</td>
<td>The values are displayed on the right in the trend window.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **TrendValueAlignment**. The data type is LONG.

### TrendValueAxis property

**Value axis - TrendValueAxis**

Defines the value axis to be used for the trend selected. Define the available value axes in the "Value axes" tab.

The attribute can be assigned dynamic properties by means of the name **TrendValueAxis**. The data type is STRING.

### TrendValueUnit property

**Unit - TrendValueUnit**

Specifies a unit for the trend type "Values" that is appended to the displayed value, e.g., "%" or "°C".

The attribute can be assigned dynamic properties by means of the name **TrendValueUnit**. The data type is STRING.
TrendVisible property

Trends - TrendVisible

The list shows all trends you created.
Select the trends to be displayed in the trend window from the list.
Click a trend entry in the list to adapt the properties and to assign axes and trend windows to the trend.
The attribute can be assigned dynamic properties by means of the name TrendVisible. The data type is BOOLEAN.

TrendWindow - TrendYAxis

TrendWindowAdd property

New - TrendWindowAdd

Creates a new trend window.
The attribute can be assigned dynamic properties by means of the name TrendWindowAdd. The data type is STRING.

TrendWindowCoarseGrid property

Main grid lines - TrendWindowCoarseGrid

Enables the display of grid lines for the main scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of grid lines for the main scale.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of grid lines for the main scale.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendWindowCoarseGrid. The data type is BOOLEAN.

TrendWindowCoarseGridColor property

Color of main scale - TrendWindowCoarseGridColor

Specifies the grid color of the main scale. Open the "Color selection" dialog by clicking the button.
The attribute can be assigned dynamic properties by means of the name TrendWindowCoarseGridColor. The data type is LONG.
TrendWindowCount property

TrendWindowCount

Defines the number of configured trend views.

The attribute can be assigned dynamic properties by means of the name TrendWindowCount. The data type is LONG.

TrendWindowFineGrid property

Secondary grid lines - TrendWindowFineGrid

Enables the display of grid lines for the secondary scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of grid lines for the secondary scale.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of grid lines for the secondary scale.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendWindowFineGrid. The data type is BOOLEAN.

TrendWindowFineGridColor property

Color of secondary scale - TrendWindowFineGridColor

Specifies the grid color of the main scale. Open the "Color selection" dialog by clicking the button.

The attribute can be assigned dynamic properties by means of the name TrendWindowFineGridColor. The data type is LONG.

TrendWindowForegroundTrendGrid property

Only for foreground trend - TrendWindowForegroundTrendGrid

Enables the display of grid lines only for the foreground trend in the trend window.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of grid lines for the foreground trend in the trend window.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Enables the display of grid lines for all trends in the trend window.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendWindowForegroundTrendGrid. The data type is BOOLEAN.
TrendWindowGridInTrendColor property

*Use trend color - TrendWindowGridInTrendColor*

Sets the trend color for the visualization of the grid lines for the main scale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The grid is displayed in the trend color.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The grid is displayed with the color set in the &quot;Color&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendWindowGridInTrendColor`. The data type is BOOLEAN.

TrendWindowHorizontalGrid property

*For X axis - TrendWindowVerticalGrid*

Enables the display of horizontal grid lines.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The display of horizontal grid lines is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The display of horizontal grid lines is disabled.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendWindowHorizontalGrid`. The data type is BOOLEAN.

TrendWindowIndex property

**TrendWindowIndex**

References a configured trend view. Using this attribute you can assign the values of other attributes to a specific trend view.

Values between 0 and "TrendWindowIndex" minus 1 are valid for "TrendWindowCount". Attribute "TrendWindowCount" defines the number of trend views configured.

The "TrendWindowIndex" attribute can be assigned dynamic properties by means of attribute `TrendWindowRepos`. The data type is LONG.

TrendWindowName property

**Object name - TrendWindowName**

Defines the name of the trend window selected.

The "TrendWindowName" attribute can be assigned dynamic properties by means of attribute `TrendWindowRename`. The data type is STRING.
TrendWindowRemove property

Remove - TrendWindowRemove
Removes the selected trend window from the list.
The attribute can be assigned dynamic properties by means of the name TrendWindowRemove. The data type is STRING.

TrendWindowRename property

TrendWindowRename
Renames a trend view which is referenced by means of "TrendWindowIndex" attribute.
The attribute can be assigned dynamic properties by means of the name TrendWindowRename. "TrendWindowRename" also sets a dynamic attribute "TrendWindowName". The data type is STRING.

TrendWindowRepos property

Up/Down - TrendWindowRepos
Changes the sorting order of the trend windows. "Up" and "Down" move the selected trend up or down in the list.
The sorting order in the list defines the position in the Control. The first trend window is displayed at the last position, while the last is displayed at the top position.
The attribute can be assigned dynamic properties by means of the name TrendWindowRepos. The data type is LONG.

TrendWindowRulerColor property

Ruler color - TrendWindowRulerColor
Specifies the ruler color. Open the "Color selection" dialog by clicking the button.
The color can be configured and displayed if "1 - graphic" is set for display of the ruler or "TrendWindowRulerStyle".
The attribute can be assigned dynamic properties by means of the name TrendWindowRulerColor. The data type is LONG.

TrendWindowRulerLayer property

Ruler layer - TrendWindowRulerLayer
Defines the representation layer of a ruler in the trend window.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Under grid</td>
<td>The ruler is visualized on a layer under the grid.</td>
</tr>
<tr>
<td>1</td>
<td>Between grid and trend</td>
<td>The ruler is positioned on top of the trend and under the grid.</td>
</tr>
<tr>
<td>2</td>
<td>On top of trend</td>
<td>The ruler is positioned on top of the trend.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendWindowRulerLayer`. The data type is LONG.

**TrendWindowRulerStyle property**

**Ruler - TrendWindowRulerStyle**

Defines the appearance of the ruler.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Simple</td>
<td>The ruler is displayed as basic black line.</td>
</tr>
<tr>
<td>1</td>
<td>Graphic</td>
<td>The ruler is displayed based on the &quot;color&quot; and &quot;weight&quot; configured.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `TrendWindowRulerStyle`. The data type is LONG.

**TrendWindowRulerWidth property**

**Ruler width - TrendWindowRulerWidth**

Defines the width of the ruler in pixels.

The width can be configured and displayed if "1 - graphic" is set for display of the ruler or "TrendWindowRulerStyle".

The attribute can be assigned dynamic properties by means of the name `TrendWindowRulerWidth`. The data type is LONG.

**TrendWindowSpacePortion property**

**Proportional area - TrendWindowSpacePortion**

Specifies the proportion of the trend widow to be used for the selected curve.

The attribute can be assigned dynamic properties by means of the name `TrendWindowSpacePortion`. The data type is LONG.
TrendWindowStatisticRulerColor property

Color of ruler for statistics area - TrendWindowStatisticRulerColor

Specifies the color of the ruler for the statistics area. The button opens the "Color selection" dialog to select the color.

The color can be configured and displayed if "1 - graphic" is set for display of the ruler for the statistics area or "TrendWindowStatisticRulerStyle".

The attribute can be assigned dynamic properties by means of the name TrendWindowStatisticRulerColor. The data type is LONG.

TrendWindowStatisticRulerStyle property

Ruler for statistics area - TrendWindowStatisticRulerStyle

Enables the display of a ruler for defining the statistics area.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Simple</td>
<td>The ruler is displayed as basic black line.</td>
</tr>
<tr>
<td>1</td>
<td>Graphic</td>
<td>The ruler is displayed based on the &quot;color&quot; and &quot;weight&quot; configured.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name TrendWindowStatisticRulerStyle. The data type is LONG.

TrendWindowStatisticRulerWidth property

Width of ruler for statistics area - TrendWindowStatisticRulerWidth

Defines the width of the ruler for the statistics area in pixels.

The width of the ruler can be configured and displayed if "1 - graphic" is set for display of the ruler for the statistics area or "TrendWindowStatisticRulerStyle".

The attribute can be assigned dynamic properties by means of the name TrendWindowStatisticRulerWidth. The data type is LONG.

TrendWindowVerticalGrid property

for Y axis - TrendWindowVerticalGrid

Enables the display of vertical grid lines.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The display of vertical grid lines is enabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The display of vertical grid lines is disabled.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name TrendWindowVerticalGrid. The data type is BOOLEAN.

**TrendWindowVisible property**

*Trend window - TimeAxisTrendWindow*

The list shows all trend windows you created.

Select the trend windows to be displayed in the control from the list.

Click a list entry to adapt the ruler and grid line properties.

The attribute can be assigned dynamic properties by means of the name TrendWindowVisible. The data type is BOOLEAN.

**TrendXAxis property**

*X axis - TrendXAxis*

Defines the X axis to be used for the trend selected. Define the available X axes in the "X Axes" tab.

The attribute can be assigned dynamic properties by means of the name TrendXAxis. The data type is STRING.

**TrendYAxis property**

*Y axis - TrendYAxis*

Defines the Y axis to be used for the trend selected. Define the available Y axes in the "Y Axes" tab.

The attribute can be assigned dynamic properties by means of the name TrendYAxis. The data type is STRING.

**Type**

**Type Property**

*Description*

Reads out the object type, e.g. "Rectangle", "Circle" or "Line".

The object type is returned as a string. Read only

A special ID is returned as the type for all the graphic elements provided by WinCC. It can be found under the topic "Type Identification in VBS" in the individual descriptions of the WinCC Object Types.
Special feature

In the case of non-WinCC controls and OLE objects, the version-independent ProgID is returned as the type.

It is possible to determine the version-dependent ProgID or "User friendly Name" from the ProgID: In the following example, "Control1" is a control embedded in the picture which already returns the version-independent ProgID as a result of the Type property.

Note

Since not every Control has a version-dependent ProgID, an error handling measure should be integrated to query the version-dependent ProgID or UserFriendlyName. If no error handling is used, the code is terminated immediately without any result when no ProgID is found.

Determine the version-dependent ProgID as follows:

```
'VBS91
Dim objControl
Dim strCurrentVersion
Set objControl = ScreenItems("Control1")
strCurrentVersion = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & "\CurVer\")
MsgBox strCurrentVersion
```

Note

In order that example above works, a multimedia control should be inserted in the picture.

Determine the User Friendly Name as follows:

```
'VBS92
Dim objControl
Dim strFriendlyName
Set objControl = ScreenItems("Control1")
strFriendlyName = CreateObject("WScript.Shell").RegRead("HKCR\" & objControl.Type & "\")
MsgBox strFriendlyName
```

Note

In order that example above works, a multimedia control should be inserted in the picture.
Example:

The following example displays the type for all objects in the picture "NewPDL1":

```vbs
'VBS93
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim lngAnswer
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    strName = objScreen.ScreenItems(lngIndex).ObjectName
    Set objScrItem = objScreen.ScreenItems(strName)
    lngAnswer = MsgBox(objScrItem.Type, vbOKCancel)
    If vbCancel = lngAnswer Then Exit For
Next
```

See also

- ScreenItem Object (Page 125)
- Object types of the ScreenItem object (Page 143)

TypeAlarmHigh Property

**Description**

TRUE, when the upper limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also

- ScreenItem Object (Page 125)
- Bar (Page 174)

TypeAlarmLow Property

**Description**

TRUE, when the lower limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
See also

Bar (Page 174)
ScreenItem Object (Page 125)

**TypeLimitHigh4 Property**

**Description**

TRUE, when the "Reserve 4" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

**TypeLimitHigh5 Property**

**Description**

TRUE, when the "Reserve 5" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)

**TypeLimitLow4 Property**

**Description**

TRUE, when the "Reserve 4" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also

Bar (Page 174)
ScreenItem Object (Page 125)
TypeLimitLow5 Property

Description
TRUE, when the "Reserve 5" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

TypeToleranceHigh Property

Description
TRUE, when the "Tolerance high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

TypeToleranceLow Property

Description
TRUE, when the "Tolerance low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

TypeWarningHigh Property

Description
TRUE, when the "Warning high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
TypeWarningLow Property

Description
TRUE, when the "Warning low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

See also
- Bar (Page 174)
- ScreenItem Object (Page 125)

UnitColor Property

Description
Defines the text color for the names of the unit of measurement. LONG write-read access.

See also
- WinCC Gauge Control (Page 254)
- ScreenItem Object (Page 125)

UnitFont Property

Description
Controls the display of the labeling for the unit of measurement. Read only access.
The following properties can be set:
- Font
- Font Style
- Font Size
- "Strikethrough" effect
- "Underline" effect

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

UnitOffset Property

Description

This attribute defines the distance of the text for the unit of measurement in relation to the top edge of the object. The text can only be positioned along the vertical diameter of the graduated scale disk. The value of the property is related to the height of the object and is measured from the top edge of the object to the base of the text.

The value range is 0 to 1.

0: The base of the text is at the top limit of the object. The text is no longer visible because it is outside the object.

1: The base of the text is at the bottom limit of the object.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

UnitText Property

Description

Defines the text for the unit of measurement. Write/Read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

UnselBGColor Property

Description

Defines or returns the background color of entries in the text list object which are not selected. LONG write-read access.
UnselTextColor Property

Description
Defines or returns the color of the text for entries in the text list object which are not selected. LONG write-read access.

See also
Text list (Page 196)
ScreenItem Object (Page 125)

UpdateCycle Property

Description
Returns the type and frequency of updating the picture window in Runtime. Read only access.

See also
Picture Window (Page 179)
ScreenItem Object (Page 125)

UpperLimit Property

Description
TRUE, when the "UpperLimitColor" specification is to be used in order to identify the tag values (from a trend referenced via "Index") which lie above the value defined in "UpperLimitValue". BOOLEAN write-read access.

See also
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)
UpperLimitColor Property

Description
Defines the color to be used in order to identify the tag values (from a trend referenced via "Index") which lie above the value defined in "UpperLimitValue". Whether the information is evaluated is dependent on the value of the "UpperLimit" property. The color is defined as an RGB value. LONG write-read access.

See also
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
ScreenItem Object (Page 125)

UpperLimitTagName Property

Description
This defines the upper limit of the trend range, which is automatically taken from the variable properties configured in PCS 7. Write/Read access.

UpperLimitValue Property

Description
Tag values (from a trend referenced via "Index") which lie above the value defined by "UpperLimitValue" are identified by the color specified in "UpperLimitColor". Whether the information is evaluated is dependent on the value of the "UpperLimit" property.

See also
ScreenItem Object (Page 125)
WinCC Online Table Control (before WinCC V7) (Page 292)
WinCC Online Trend Control (before WinCC V7) (Page 295)
WinCC Function Trend Control (before WinCC V7) (Page 288)
Us

UseColumnBackColor property

Use column color / background - UseColumnBackColor

Specifies the settings to be activated for the background colors of columns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The background color settings are active in the &quot;Time columns&quot; or &quot;TimeColumnBackColor&quot; tabs and in the &quot;Value columns&quot; or &quot;ValueColumnBackColor&quot; tabs.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The background color settings are active in the &quot;Display&quot; tab.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseColumnBackColors. The data type is BOOLEAN.

UseColumnForeColor property

Use column color / font - UseColumnForeColor

Defines the active font color settings for the columns.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The font color color settings are active in the &quot;Time columns&quot; or &quot;TimeColumnForeColor&quot; tabs and in the &quot;Value columns&quot; or &quot;ValueColumnForeColor&quot; tabs.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The font color settings are active in the &quot;Display&quot; tab.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseColumnForeColors. The data type is BOOLEAN.

UseMessageColor property

Show message colors - UseMessageColor

Sets the outputs of messages with colors as agreed by handshake.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The message colors are displayed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The message colors are not displayed. Instead, the color settings defined for the table content are activated on the &quot;Display&quot; tab.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseMessageColor. The data type is BOOLEAN.
UseOnlineTags Property

Description
This defines whether or not the variable properties configured in PCS 7 are applied as trend parameters. Write/Read access.

UseRangeSubstitutes Property

Description
TRUE, if a separate scaling of the value axis is displayed for the trends in Trend Control. BOOLEAN write-read access.

UserData-Property

Description
Contains the value that is to be transferred to the VB script while running a customized menu item or icon. STRING (write-read access)

Example:
Use the "User data" field in the "Menus and Toolbars" editor to apply a parameter to the procedure
The following example shows the "ActivateScreen" procedure that executes the picture change. Enter the picture name in the "User Data" field:

Sub ActivateScreen (ByVal Item)
Dim objScreen
Dim strScreenName
' "UserData" contains the screen name specified
' in editor menus and toolbars.
strScreenName = Item.Userdata
HMIRuntime.BaseScreenName = strScreenName
End Sub

UserName property

Description
Returns the name of the user who triggered the alarm object.
See also

Alarms object (list) (Page 110)

UserValue1 Property

Description

Defines or returns a value. The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

See also

ScreenItem Object (Page 125)
Group Display (Page 194)

UserValue2 Eigenschaft

Description

Defines or returns a value. The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)

UserValue3 Property

Description

Defines or returns a value. The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

See also

Group Display (Page 194)
ScreenItem Object (Page 125)
UserValue4 Property

Description
Defines or returns a value.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

See also
Group Display (Page 194)
ScreenItem Object (Page 125)

UseSelectedTitleColor property

Selection color - UseSelectedTitleColor
Specifies whether to use a selection color for the headers of selected table cells.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>A selection color is used. The &quot;Background&quot; or &quot;SelectedTitleColor&quot; and &quot;Font&quot; or &quot;SelectedTitleForeColor&quot; settings are active in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Selection color is not used. The &quot;Background&quot; and &quot;Font&quot; settings are disabled in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseSelectedTitleColor. The data type is BOOLEAN.

UseSourceBackColors property

Apply background colors - UseSourceBackColors
Sets the background color derived from the control defined in the "Source" field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The background color from the interconnected control is used.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The background color from the interconnected control is not used. The settings on the &quot;Layout&quot; tab are used.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseSourceBackColors. The data type is BOOLEAN.
UseSourceForeColors property

Apply font colors - UseSourceForeColors

Sets the font colors derived from the control defined in the "Source" field.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The font color of the interconnected control is activated.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The font color from the connected control is not used. The settings on the &quot;Layout&quot; tab are used.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseSourceForeColors. The data type is BOOLEAN.

UseTableColor2 property

Row Color 2 - UseTableColor2

Specifies whether to use a second row color for the representation of the table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>&quot;Row color 2&quot; and &quot;Row color 1&quot; are used alternately.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The &quot;Row color 1&quot; settings are used for all rows.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseTableColor2. The data type is BOOLEAN.

UseTrendNameAsLabel property

UseTrendNameAsLabel

Sets the "TrendName" or "TrendLabel" attribute for labeling the trend in Runtime.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Sets the &quot;TrendName&quot; attribute for labeling the trend in Runtime.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Sets the &quot;TrendLabel&quot; attribute for labeling the trend in Runtime.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name UseTrendNameAsLabel. The data type is BOOLEAN.
1.14.4.21  V

Val - ValueAxis

Value Property

Description of Tag Object

Displays the value of the tags at the last read access or the value written or to be written. Value represents the value of a tag. After calling in the "Read" method, the tag value read is returned. Before writing, the new tag value required can be assigned to the property. After calling in the "Write" method, the property contains the value last written.

VARIANT (write-read access)

Example:

The following example writes a new value in the "Tag1" tag:

'VBS94
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 50
objTag.Write

Description of WinCC Gauge Control

Defines the value to which the pointer points. Value Range: "ValueMin" to "ValueMax".

Description of DataItem Object

Returns a value copy or object reference. Furthermore, an already added value can be changed via the value property.
Example:

The example shows how to add a value to the list, and how to output it as a trace. After that, the value is changed, output again and then removed. It make sense to perform this in several different actions.

'VBS198
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet("motor1").Value = 55
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")

Note

For object references it must be ascertained that objects are multiread-enabled.

See also

WinCC Gauge Control (Page 254)
Write Method (Page 813)
Read Method (Page 783)
Tag Object (Page 137)
DataItem Object (Page 113)
ProcessValues Object (List) (Page 124)

ValueAxisAdd property

New - ValueAxisAdd

Creates a new value axis.

The attribute can be assigned dynamic properties by means of the name ValueAxisAdd. The data type is STRING.

ValueAxisAlign property

Alignment - ValueAxisAlign

Specifies the mode of alignment of a selected value axis.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The selected value axis is displayed on left side of the trend or diagram.</td>
</tr>
<tr>
<td>1</td>
<td>right</td>
<td>The selected value axis is displayed on right side of the trend or diagram.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueAxisAlign`. The data type is LONG.

### ValueAxisAutoPrecisions property

**Decimal places automatic - ValueAxisAutoPrecisions**

Enables automatic setting of the decimal precision.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The decimal precision is defined automatically. The value in the &quot;Decimal places&quot; or &quot;ValueAxisPrecisions&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value in the &quot;Decimal places&quot; or &quot;ValueAxisPrecisions&quot; field is active.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueAxisAutoPrecisions`. The data type is BOOLEAN.

### ValueAxisAutoRange property

**Value range automatic - ValueAxisAutoRange**

Enables automatic calculation of the range of values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The range of values is calculated automatically.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The range of values is calculated based on the values configured in the &quot;from&quot; and &quot;to&quot; or &quot;ValueAxisBeginValue&quot; and &quot;ValueAxisEndValue&quot; fields.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueAxisAutoRange`. The data type is BOOLEAN.

### ValueAxisBarWindow property

**Diagram window - ValueAxisBarWindow**

Specifies the diagram window in which the selected value axis will be used. You can specify the available diagram window in the "Diagram window" tab or using "BarWindowAdd".

The attribute can be assigned dynamic properties using the name `ValueAxisBarWindow`. The data type is STRING.
ValueAxisBeginValue property

Value range from - ValueAxisBeginValue

Specifies the start value of the value axis selected. You can configure the value if the "Automatic" option is disabled or "ValueAxisAutoRange" is "FALSE".

The attribute can be assigned dynamic properties by means of the name ValueAxisBeginValue. The data type is DOUBLE.

ValueAxisColor property

Value axis color - ValueAxisColor

Specifies the color of the time axis. The button opens the "Color selection" dialog to select the color.

The setting is only active if the "Use trend color" or "Use diagram color" option is disabled or if "ValueAxisInTrendColor" is "FALSE".

The attribute can be assigned dynamic properties by means of the name ValueAxisColor. The data type is LONG.

ValueAxisCount property

ValueAxisCount

Defines the number of value axes configured.

The attribute can be assigned dynamic properties by means of the name ValueAxisCount. The data type is LONG.

ValueAxisEndValue property

Value range to - ValueAxisEndValue

Specifies the end value of the value axis selected. You can configure the value if the "Automatic" option is disabled or "ValueAxisAutoRange" is "FALSE".

The attribute can be assigned dynamic properties by means of the name ValueAxisEndValue. The data type is DOUBLE.
ValueAxisExponentialFormat property

Exponential notation - ValueAxisExponentialFormat

Sets exponential notation for the display of values of a value axis selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The values are displayed with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The values are displayed with decimal notation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name **ValueAxisExponentialFormat**. The data type is BOOLEAN.

ValueAxisInBarColor property

In diagram color - ValueAxisInBarColor

Specifies whether the selected value axis will be displayed in the diagram color. If there is more than one diagram in the diagram window, the color of the first diagram will be used. You specify the order of the diagrams on the "Diagrams" tab.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The selected value axis is displayed in the diagram color. The setting in the &quot;Color&quot; or &quot;ValueAxisColor&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value axis selected is displayed in the color set in the &quot;Color&quot; or &quot;ValueAxisColor&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties using the name **ValueAxisInBarColor**. The data type is BOOLEAN.

ValueAxisIndex property

ValueAxisIndex

References a value axis. Using this attribute you can assign the values of other attributes to a specific value axis.

Values between 0 and "ValueAxisCount" minus 1 are valid for "ValueAxisIndex". Attribute "ValueAxisCount" defines the number of value axes configured.

The "ValueAxisIndex" attribute can be assigned dynamic properties by means of attribute **ValueAxisRepos**. The data type is LONG.
ValueAxisInTrendColor property

Use trend color - ValueAxisInTrendColor

Sets the trend color for displaying the value axis selected. The color of the first trend is activated if several trends are displayed in the trend window. Define the order of trends on the "Trends" tab.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The selected value axis is displayed in the trend color. The setting in the &quot;Color&quot; or &quot;ValueAxisColor&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value axis selected is displayed in the color set in the &quot;Color&quot; or &quot;ValueAxisColor&quot; field.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ValueAxisInTrendColor. The data type is BOOLEAN.

ValueAxisLabel property

Label - ValueAxisLabel

Specifies the label of a value axis selected.

The attribute can be assigned dynamic properties by means of the name ValueAxisLabel. The data type is STRING.

ValueAxisName property

Object name - ValueAxisName

Specifies the name of a value axis selected.
The "ValueAxisName" attribute can be assigned dynamic properties by means of attribute ValueAxisRename. The data type is STRING.

ValueAxisPrecisions property

Decimal places - ValueAxisPrecisions

规定的十进制精度用于显示选择的数值轴。该数值可以配置且在运行时有效，如果"Automatic"选项禁用或“ValueAxisAutoPrecisions”为"FALSE"。

The attribute can be assigned dynamic properties by means of the name ValueAxisPrecisions. The data type is SHORT.

ValueAxisRemove property

Remove - ValueAxisRemove

移除选择的数值轴从列表。

The attribute can be assigned dynamic properties by means of the name ValueAxisRemove. The data type is STRING.

ValueAxisRename property

ValueAxisRename

renames a value axis which is referenced by means of "ValueAxisIndex" attribute.

The attribute can be assigned dynamic properties by means of the name ValueAxisRename. "ValueAxisRename" also sets a dynamic attribute "ValueAxisName". The data type is STRING.

ValueAxisRepos property

Up/Down - ValueAxisRepos

Changes the order of value axes. "Up" and "Down" move the value axis selected up or down in the list.

The list order determines the value axis position in the trend window or diagram window in runtime. If the orientation is the same and the value axis of the list is further above, the value axis is shown at a more remote position of the curve or diagram.

The attribute can be assigned dynamic properties by means of the name ValueAxisRepos. The data type is LONG.
ValueAxisScalingType property

Scaling - ValueAxisScalingType

Specifies the scaling mode for a selected value axis.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Linear</td>
<td>Enables linear scaling of a value axis selected.</td>
</tr>
<tr>
<td>1</td>
<td>Logarithmic</td>
<td>Enables logarithmic scaling of a value axis selected.</td>
</tr>
<tr>
<td>2</td>
<td>Logarithmically negated</td>
<td>Enables scaling of a selected value axis with logarithmic negation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ValueAxisScalingType. The data type is LONG.

ValueAxisTrendWindow property

Trend window - ValueAxisTrendWindow

Specifies the trend window for displaying the value axis selected. Define the available trend windows in the "Trend window" tab.

The attribute can be assigned dynamic properties by means of the name ValueAxisTrendWindow. The data type is STRING.

ValueAxisVisible property

Value axes - ValueAxisVisible

The list shows all value axes you created. Click a value axis entry in the list to adapt the properties and to assign a trend window or diagram window to the value axis.

Select the value axes in the list that you want to display in the trend windows or diagram windows.

The attribute can be assigned dynamic properties by means of the name ValueAxisVisible. The data type is BOOLEAN.

ValueColumn - Vi

ValueColumnAdd property

New - ValueColumnAdd

Creates a new value column.
The attribute can be assigned dynamic properties by means of the name `ValueColumnAdd`. The data type is STRING.

**ValueColumnAlign property**

**Alignment - ValueColumnAlign**

Defines the mode of alignment of a selected value column.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The selected value column is displayed on the left.</td>
</tr>
<tr>
<td>1</td>
<td>Centered</td>
<td>The selected value column is aligned to center.</td>
</tr>
<tr>
<td>2</td>
<td>right</td>
<td>The selected value column is displayed on the right.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueColumnAlign`. The data type is LONG.

**ValueColumnAlignment Property**

**Description**

The "Index" property references a pair of columns. "ValueColumnAlignment" defines the alignment of the tag value for this column pair.

- 0: Tag values are entered aligned left.
- 1: Tag values are entered centered.
- 2: Tag values are entered aligned right.

**See also**

WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

**ValueColumnAutoPrecisions property**

**Automatic - ValueColumnAutoPrecisions**

Enables automatic setting of the decimal precision.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The decimal precision is defined automatically. The value in the &quot;Decimal places&quot; or &quot;ValueColumnPre-</td>
</tr>
<tr>
<td>FALSE</td>
<td>cisions&quot; field is disabled.</td>
</tr>
</tbody>
</table>
The attribute can be assigned dynamic properties by means of the name `ValueColumnAutoPrecisions`. The data type is BOOLEAN.

**ValueColumnBackColor property**

**Background color - ValueColumnBackColor**

Specifies the background color of the value column selected. Use the button to open the "Color selection" dialog.

The setting is only active if the "Background color" option is set or "UseColumnBackColor" is "TRUE" in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name `ValueColumnBackColor`. The data type is LONG.

**ValueColumnCaption property**

**Description - ValueColumnCaption**

Defines the label of the value column selected.

The attribute can be assigned dynamic properties by means of the name `ValueColumnCaption`. The data type is STRING.

**ValueColumnCount property**

**ValueColumnCount**

Defines the number of value columns configured.

The attribute can be assigned dynamic properties by means of the name `ValueColumnCount`. The data type is LONG.

**ValueColumnExponentialFormat property**

**Exponential notation - ValueColumnExponentialFormat**

Sets exponential notation for the display of values of a value column selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Display with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Display with decimal notation.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueColumnExponentialFormat`. The data type is BOOLEAN.
ValueColumnForeColor property

Font color - ValueColumnForeColor

Specifies the font color of the value column selected. Use the button to open the "Color selection" dialog.

The setting is only active if the "Font color" option is set or "UseColumnForeColor" is "TRUE" in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name ValueColumnForeColor. The data type is LONG.

ValueColumnHideText property

ValueColumnHideText

Sets text format for displaying the content of a value column.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is not displayed in text format.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is displayed in text format.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ValueColumnHideText. The data type is BOOLEAN.

ValueColumnHideTitleText property

ValueColumnHideTitleText

Sets text format for displaying the value column header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is not displayed in text format.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is displayed in text format.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ValueColumnHideTitleText. The data type is BOOLEAN.

ValueColumnIndex property

ValueColumnIndex

References a configured value column. Using this attribute you can assign the values of other attributes to a specific value column.

Values between 0 and "ValueColumnCount" minus 1 are valid for "ValueColumnIndex". Attribute "ValueColumnCount" defines the number of value columns configured.
The "ValueColumnIndex" attribute can be assigned dynamic properties by means of attribute ValueColumnRepos. The data type is LONG.

**ValueColumnLength property**

**Length in characters - ValueColumnLength**

Specifies the width of a selected value column.

The attribute can be assigned dynamic properties by means of the name ValueColumnLength. The data type is LONG.

**ValueColumnName property**

**Object name - ValueColumnName**

 Specifies the name of a selected value column.

The "ValueColumnName" attribute can be assigned dynamic properties by means of attribute ValueColumnRename. The data type is STRING.

**ValueColumnPrecisions property**

**Decimal places - ValueColumnPrecisions**

 Specifies the decimal precision for displaying the data of a value column selected. The value can be entered if the "Automatic" option is disabled or "ValueColumnAutoPrecisions" is "FALSE".

The attribute can be assigned dynamic properties by means of the name ValueColumnPrecisions. The data type is SHORT.

**ValueColumnProvider property**

**Data source - ValueColumnProvider**

 Specifies the data source for a selected value column.

The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Archive tags</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
<tr>
<td>2</td>
<td>Online tags</td>
<td>Data source with online tags derived from tag management.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name ValueColumnProvider. The data type is LONG.
ValueColumnProviderCLSID property

ValueColumnProviderCLSID

Indicates the data source of the value column selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>{416A09D2-8B5A-11D2-8B81-006097A45D48}</td>
<td>Data source with archive tags of a process value archive.</td>
</tr>
<tr>
<td>{A3F69593-8AB0-11D2-A440-00A0C9DDB64E}</td>
<td>Data source with online tags derived from tag management.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueColumnProviderCLSID`. The data type is STRING.

ValueColumnRemove property

Remove - ValueColumnRemove

Removes the selected value column from the list.

The attribute can be assigned dynamic properties by means of the name `ValueColumnRemove`. The data type is STRING.

ValueColumnRename property

ValueColumnRename

Renames a value column which is referenced by means of "ValueColumnIndex" attribute.

The attribute can be assigned dynamic properties by means of the name `ValueColumnRename`. "ValueColumnRename" also sets a dynamic attribute "ValueColumnName". The data type is STRING.

ValueColumnRepos property

Up/Down - ValueColumnRepos

Changes the sorting order of the value columns. "Up" and "Down" move the value column selected up or down in the list.

The sorting order in the list determines the order of value columns after the time column if several value columns are assigned to the same time column. Higher positions of the value column in the list moves it to closer proximity towards the time column.

You change the order of time columns and their assigned value columns in the "Time columns" tab.

The attribute can be assigned dynamic properties by means of the name `ValueColumnRepos`. The data type is LONG.
**ValueColumnSelectTagName property**

**ValueColumnSelectTagName**

Opens a dialog for selecting the tag name for the data source of the value column in WinCC OnlineTableControl. Programmers can set this attribute to allow users to select a tag name by means of a button, for example.

The attribute can be assigned dynamic properties by means of the name `ValueColumnSelectTagName`. The data type is BOOLEAN.

**ValueColumnShowIcon property**

**ValueColumnShowIcon**

Enables the display of value column contents as icon.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The content is visualized as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The content is not visualized as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueColumnShowIcon`. The data type is BOOLEAN.

**ValueColumnShowTitleIcon property**

**ValueColumnShowTitleIcon**

Enables display of the value column header as icon.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The header is displayed as icon.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The header is not displayed as icon.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `ValueColumnShowTitleIcon`. The data type is BOOLEAN.

**ValueColumnSort property**

**ValueColumnSort**

Defines the sorting order of the value column referenced in "ValueColumnIndex".
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>No sorting</td>
</tr>
<tr>
<td>1</td>
<td>Ascending</td>
<td>Ascending order, starting at the lowest value.</td>
</tr>
<tr>
<td>2</td>
<td>Descending</td>
<td>Descending order, starting at the highest value.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name \texttt{ValueColumnSort}. The data type is \texttt{LONG}.

\textbf{ValueColumnSortIndex property}

\textbf{ValueColumnSortIndex}

Defines the sorting order of the value column referenced in "ValueColumnIndex". The sorting criterion is removed from "ValueColumnSort" if you set a "0" value.

The attribute can be assigned dynamic properties by means of the name \texttt{ValueColumnSortIndex}. The data type is \texttt{LONG}.

\textbf{ValueColumnState property}

\textbf{ValueColumnState}

Displays the data connection status of a selected value column in Runtime.

The attribute can be assigned dynamic properties by means of the name \texttt{ValueColumnState}. The data type is \texttt{LONG}.

\textbf{ValueColumnTagName property}

\textbf{Tag name - ValueColumnTagName}

Displays the name of connected tags. You can change the tag connection using the selection button.

The attribute can be assigned dynamic properties by means of the name \texttt{ValueColumnTagName}. The data type is \texttt{STRING}.

\textbf{ValueColumnTimeColumn property}

\textbf{Time column - ValueColumnTimeColumn}

Specifies the time column for displaying the value column selected. Define the available time columns in the "Time columns" tab.

The attribute can be assigned dynamic properties by means of the name \texttt{ValueColumnTimeColumn}. The data type is \texttt{STRING}. 
ValueColumnVisible property

Value columns - ValueColumnVisible

The list shows all value columns you created. Click a value column entry in the list to adapt the properties, to assign the value column, and to define the data connection.

Select the value columns to be displayed in the table from the list. Value columns are displayed if interconnected with a time column.

The attribute can be assigned dynamic properties by means of the name ValueColumnVisible. The data type is BOOLEAN.

ValueMax Property

Description

Defines the value at the end of the scale. Write/Read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

ValueMin Property

Description

Defines the value at the start of the scale. Write/Read access.

See also

WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

Variable Property

Description

The "Index" property references a pair of columns. "Tag" defines the name of the tag which should be connected to this column pair.
See also

WinCC Online Table Control (before WinCC V7) (Page 292)
ScreenItem Object (Page 125)

VerticalGridLines property

Vertical - VerticalGridLines

Enables the display of vertical dividers.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Enables the display of vertical dividers.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Disables the display of vertical dividers.</td>
</tr>
</tbody>
</table>

The attribute can be assigned dynamic properties by means of the name `VerticalGridLines`. The data type is BOOLEAN.

Visible Property

Description

witches an object visible or invisible or issues a corresponding value:

- TRUE : Object is visible
- FALSE : Object is invisible

VARIANT_BOOL (write-read access)

Example:

The following example sets all the objects in the picture "NewPDL1" to invisible:

'VBS95
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    strName = objScreen.ScreenItems(lngIndex).ObjectName
    Set objScrItem = objScreen.ScreenItems(strName)
    objScrItem.Visible = False
Next
See also
ScreenItem Object (Page 125)
Layer Object (Page 120)
HMIRuntime Object (Page 118)

1.14.4.22 W

Warning Property

Description
Defines the start of the "Warning zone" as a scale value. Write/Read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

WarningColor Property

Description
Defines the color of the "Warning zone" on the scale. LONG write-read access.

See also
WinCC Gauge Control (Page 254)
ScreenItem Object (Page 125)

WarningHigh Property

Description
Defines or returns the upper limit value for "Warning High". In order that the limit value is monitored, the "CheckWarningHigh" property must be set to TRUE. The display on reaching the limit value and the type of evaluation are defined by means of the "ColorWarningHigh" and "TypeWarningHigh" properties.
WarningLow Property

Description
Defines or returns the lower limit value for "Warning Low".
In order that the limit value is monitored, the "CheckWarningLow" property must be set to TRUE.
The display on reaching the limit value and the type of evaluation are defined by means of the "ColorWarningLow" and "TypeWarningLow" properties.

See also
Bar (Page 174)
ScreenItem Object (Page 125)

Width Property

Description
Sets or outputs the width of an object in pixels.

LONG
Example:

The following example doubles the width of all objects in the pictures "NewPDL1" whose name begins with "Button":

```vbs
'VBS96
Dim objScreen
Dim cmdButton
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    'Get all "Buttons"
    strName = objScreen.ScreenItems(lngIndex).ObjectName
    If "Button" = Left(strName, 6) Then
        Set cmdButton = objScreen.ScreenItems(strName)
        cmdButton.Width = cmdButton.Width * 2
    End If
Next
```

See also

- Height Property (Page 443)
- ScreenItem Object (Page 125)

WinCCStyle property

Description

Defines the style in which the object is displayed.

- User Defined
  Shows the object according to the respective settings.
- Global
  Shows the object in a globally defined design.
- Windows Style
  Shows the object in Windows style.

WindowBorder Property

Description

TRUE, when the window is displayed with borders in Runtime. Read only access.
See also

- Picture Window (Page 179)
- Application Window (Page 173)
- ScreenItem Object (Page 125)

WindowPositionMode property

Description

Defines the position and scaling of the picture window on the screen. It is only effective if the "Independent window" attribute is set to TRUE.

- Standard: The picture window is positioned in its original size in the configured position on the screen.
- Center: The picture window is positioned in its original size, centered on the screen.
- Maximize: The picture window is scaled to the size of the screen.

WindowsStyle property

Description

Defines whether the object is displayed in the Windows style of WinCC version 6.2. It can only be selected if "WinCC Classic" is chosen as the current design.

- TRUE if the object is displayed in the Windows style of WinCC version 6.2.
- FALSE if the object is not displayed in the Windows style of WinCC version 6.2.

WindowsStyle Property

Description

- TRUE, when the object complies with the general Windows style (e.g. gray buttons without borders). BOOLEAN write-read access. Note:
  - When this property is set to "True", the properties which do not comply with the Windows style are ignored (e.g. "BorderWidth").
  - On the other hand, the definition of a "BorderWidth" or a background color other than gray causes "WindowsStyle" to receive the value "False".
  - Exceptions here are the flash attributes: The definition of flash attributes does not automatically lead to the deactivation of the "WindowsStyle" attribute.
See also

Slider (Page 212)
Button (Page 201)
ScreenItem Object (Page 125)

WindowType Property

Description

Defines the use of the message window.

- 0 - Message list: shows the currently pending messages.
- 1 - Short-term archive list: shows the archived messages.
- 2 - Long-term archive list: shows the archived messages.
- 3 - Lock list: shows the currently locked messages.
- 4 - Hit list: To display the statistical information of messages.

See also

WinCC Alarm Control (before WinCC V7) (Page 286)
ScreenItem Object (Page 125)

WithAxes Property

Description

TRUE, when the scale should be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 277)
ScreenItem Object (Page 125)

WithLabels Property

Description

TRUE, when the scale labels should be displayed. BOOLEAN write-read access.
X/YAxisColor property (before WinCC V7)

Description

Use this attribute to define the color for the common X-axis. The color is defined as an RGB value. LONG write-read access.

X/YAxisAdd property

New - X/YAxisAdd

Creates a new X or Y axis.

The X axis attribute can be assigned dynamic properties by means of the name XAxisAdd.

The Y axis attribute can be assigned dynamic properties by means of the name YAxisAdd.

The data type is STRING.

X/YAxisAlign property

Alignment - X/YAxisAlign

Defines the alignment mode for a selected axis.

The following settings are available for the X axis:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bottom</td>
<td>The X axis selected is displayed below the trend.</td>
</tr>
<tr>
<td>1</td>
<td>Top</td>
<td>The X axis selected is displayed above the trend.</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name XAxisAlign.

The data type is LONG.

The following settings are available for the Y axis:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>left</td>
<td>The X axis selected is displayed on left side of the trend.</td>
</tr>
<tr>
<td>1</td>
<td>right</td>
<td>The X axis selected is displayed on right side of the trend.</td>
</tr>
</tbody>
</table>

The Y axis attribute can be assigned dynamic properties by means of the name YAxisAlign.

The data type is LONG.
**X/YAxisAutoPrecisions property**

**Decimal places automatic - X/YAxisAutoPrecisions**

Enables automatic setting of the decimal precision.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The number of decimal places is set automatically. The value in the &quot;Decimal places&quot; or &quot;X/YAxisPrecisions&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The value in the &quot;Decimal places&quot; or &quot;X/YAxisPrecisions&quot; field is active.</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name `XAxisAutoPrecisions`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisAutoPrecisions`.

The data type is BOOLEAN.

**X/YAxisAutoRange property**

**Value range automatic - X/YAxisAutoRange**

Enables automatic calculation of the value range of the axis selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The range of values is calculated automatically.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The range of values is calculated based on the values configured in the &quot;from&quot; and &quot;to&quot; or &quot;X/YAxisBeginValue&quot; and &quot;X/YAxisEndValue&quot; fields.</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name `XAxisAutoRange`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisAutoRange`.

The data type is BOOLEAN.

**X/YAxisBeginValue property**

**Value range from - X/YAxisBeginValue**

Specifies the lower range of values of the axis selected. You can configure the value if the "Automatic" option is disabled or "X/YAxisAutoRange" is "FALSE".

The X axis attribute can be assigned dynamic properties by means of the name `XAxisBeginValue`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisBeginValue`.

The data type is DOUBLE.
X/YAxisColor property

Color XY axis - X/YAxisColor

Specifies the color of the axis selected. The button opens the "Color selection" dialog to select the color.

The setting is only active if the "Use trend color" field is disabled or "X/YAxisInTrendColor" is "FALSE".

The X axis attribute can be assigned dynamic properties by means of the name XAxisColor.

The Y axis attribute can be assigned dynamic properties by means of the name YAxisColor.

The data type is LONG.

X/YAxisEndValue property

Value range to - X/YAxisEndValue

Specifies the upper range of values of the axis selected. You can configure the value if the "Automatic" option is disabled or "X/YAxisAutoRange" is "FALSE".

The X axis attribute can be assigned dynamic properties by means of the name XAxisEndValue.

The Y axis attribute can be assigned dynamic properties by means of the name YAxisEndValue.

The data type is DOUBLE.

X/YAxisExponentialFormat property

Exponential notation - X/YAxisExponentialFormat

Enables the exponential notation for visualization of a selected axis.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The values are displayed with exponential notation.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The values are displayed with decimal notation.</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name XAxisExponentialFormat.

The Y axis attribute can be assigned dynamic properties by means of the name YAxisExponentialFormat.

The data type is BOOLEAN.
**X/YAxisInTrendColor property**

**Use trend color - X/YAxisInTrendColor**

Enables the display of an axis selected in the trend color. The color of the first trend is activated if several trends are displayed in the trend window. Define the order of trends on the "Trends" tab.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The axis selected is displayed in the trend color. The setting in the &quot;Color&quot; or &quot;X/YAxisColor&quot; field is disabled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The axis selected is displayed in the color set in the &quot;Color&quot; or &quot;X/YAxisColor&quot; field.</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name **XAxisInTrendColor**.

The Y axis attribute can be assigned dynamic properties by means of the name **YAxisInTrendColor**.

The data type is BOOLEAN.

**X/YAxisLabel property**

**Label - X/YAxisLabel**

Defines the label text for a selected axis.

The X axis attribute can be assigned dynamic properties by means of the name **XAxisLabel**.

The Y axis attribute can be assigned dynamic properties by means of the name **YAxisLabel**.

The data type is STRING.

**X/YAxisName property**

**Object name - X/YAxisName**

Specifies the name of a selected axis.

Attribute "XAxisName" can be assigned dynamic properties for the X axis by means of **XAxisRename** attribute.

Attribute "YAxisName" can be assigned dynamic properties for the Y axis by means of **YAxisRename** attribute.

The data type is STRING.
X/YAxisPrecisions property

Decimal places - X/YAxisPrecisions

Specifies the decimal precision for displaying the axis selected. The value can be configured and is active in Runtime, if the "Automatic" option is disabled or "X/YAxisAutoPrecisions" is "FALSE".

The X axis attribute can be assigned dynamic properties by means of the name XAxisPrecisions.
The Y axis attribute can be assigned dynamic properties by means of the name YAxisPrecisions.
The data type is SHORT.

X/YAxisRemove property

Remove - X/YAxisRemove

Removes the selected axis from the list.
The X axis attribute can be assigned dynamic properties by means of the name XAxisRemove.
The Y axis attribute can be assigned dynamic properties by means of the name YAxisRemove.
The data type is STRING.

X/YAxisRepos property

Up/Down - X/YAxisRepos

Changes the sorting order of the axes. "Up" and "Down" move the axis selected up or down in the list.
The list order determines the axis position in the trend window. The axis output position is moved away from the trend if the axis is moved further up in the list and the orientation is the same.
The X axis attribute can be assigned dynamic properties by means of the name XAxisRepos.
The Y axis attribute can be assigned dynamic properties by means of the name YAxisRepos.
The data type is LONG.

X/YAxisScalingType property

Scaling - X/YAxisScalingType

Defines the scaling mode for a selected axis.
The following settings are available:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Linear</td>
</tr>
<tr>
<td>1</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>2</td>
<td>Logarithmically negated</td>
</tr>
</tbody>
</table>

The X axis attribute can be assigned dynamic properties by means of the name `XAxisScalingType`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisScalingType`.

The data type is LONG.

**X/YAxisTrendWindow property**

*Trend window - X/YAxisTrendWindow*

Specifies the trend window for a selected axis. Define the available trend windows in the "Trend window" tab.

The X axis attribute can be assigned dynamic properties by means of the name `XAxisTrendWindow`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisTrendWindow`.

The data type is STRING.

**X/YAxisVisible property**

*X/Y axes - X/YAxisVisible*

The list shows all axes you created. Click an axis entry in the list to adapt the properties and to assign the axis to a trend window.

Activate the axes to be displayed in the trend windows in the list.

The X axis attribute can be assigned dynamic properties by means of the name `XAxisVisible`.

The Y axis attribute can be assigned dynamic properties by means of the name `YAxisVisible`.

The data type is BOOLEAN.

**XAxisCount property**

**XAxisCount**

Defines the number of X axes configured.
The attribute can be assigned dynamic properties by means of the name \texttt{XAxisCount}. The data type is LONG.

\textbf{XAxisIndex property}

\textbf{XAxisIndex} 

References a configured X axis. Using this attribute you can assign the values of other attributes to a specific X axis.

Values between 0 and "XAxisCount" minus 1 are valid for "Index"; the attribute "XAxisCount" defines the number of configured X axes.

The "XAxisIndex" attribute can be assigned dynamic properties by means of attribute \texttt{XAxisRepos}. The data type is LONG.

\textbf{XAxisRename property}

\textbf{XAxisRename} 

Renames the X axis which is referenced by means of "XAxisIndex" attribute.

The attribute can be assigned dynamic properties by means of the name \texttt{XAxisRename}. "XAxisRename" also sets a dynamic attribute "XAxisName". The data type is STRING.

\textbf{YAxisCount property}

\textbf{YAxisCount} 

Defines the number of Y axes configured.

The attribute can be assigned dynamic properties by means of the name \texttt{YAxisCount}. The data type is LONG.

\textbf{YAxisIndex property}

\textbf{YAxisIndex} 

References a configured Y axis. Using this attribute you can assign the values of other attributes to a specific Y axis.

Values between 0 and "YAxisCount" minus 1 are valid for "Index". Attribute "YAxisCount" defines the number of configured Y axes.

The "YAxisIndex" attribute can be assigned dynamic properties by means of attribute \texttt{YAxisRepos}. The data type is LONG.
YAxisRename property

YAxisRename

Renames the Y axis which is referenced by means of "YAxisIndex" attribute. The attribute can be assigned dynamic properties by means of the name `YAxisRename`. "YAxisRename" also sets a dynamic attribute "YAxisName". The data type is STRING.

ZeroPoint Property

Description

Defines or returns the position of the zero point of the bar graph. Specify the value as a %age of the total bar height. The zero point can also be outside of the range represented. The "ScalingType" property must be set to "2" and "Scaling" to TRUE.

See also

ScreenItem Object (Page 125)
Bar (Page 174)

ZeroPointValue Property

Description

Defines the value of the zero point of the scale indicator. Defines or returns the absolute value for the zero point.

See also

Bar (Page 174)
3D Bar (Page 169)
ScreenItem Object (Page 125)

Zoom Property

Description

Sets the zoom factor within a picture or picture window or reads it out. If the indicated zoom factor is smaller than the minimum value, the zoom factor is automatically set to the minimum value. If the indicated zoom factor is larger than the minimum value, the zoom factor is automatically set to the maximum value.
The minimum value of the zoom factor is at 2%, the maximum value at 800%.
With the Screen Object the zoom factor is indicated as a numeric value and with a picture window object, it is indicated in percent.

**Example:**

The following example doubles the zoom factor of the current picture:

```vbs
'VBS97
HMIRuntime.ActiveScreen.Zoom = HMIRuntime.ActiveScreen.Zoom * 2
```

**See also**

- Picture Window (Page 179)
- Screen Object (Page 131)

### 1.14.5 Methods

#### 1.14.5.1 Methods

**Overview**

Methods, which are applied to individual objects, can be used to read out tag values for further processing or displaying diagnostics messages in Runtime.

<table>
<thead>
<tr>
<th>Available Methods in VBS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate</td>
<td>GetStatusBarElement</td>
</tr>
<tr>
<td>ActivateDynamic</td>
<td>GetStatusBarElementCollection</td>
</tr>
<tr>
<td>Add</td>
<td>GetTimeAxis</td>
</tr>
<tr>
<td>AttachDB</td>
<td>GetTimeAxisCollection</td>
</tr>
<tr>
<td>CalculateStatistic</td>
<td>GetTimeColumn</td>
</tr>
<tr>
<td>CopyRows</td>
<td>GetTimeColumnCollection</td>
</tr>
<tr>
<td>CreateTagSet</td>
<td>GetToolbarButton</td>
</tr>
<tr>
<td>CutRows</td>
<td>GetToolbarButtonCollection</td>
</tr>
<tr>
<td>DeactivateDynamic</td>
<td>GetTrend</td>
</tr>
<tr>
<td>DeleteRows</td>
<td>GetTrendCollection</td>
</tr>
<tr>
<td>DetachDB</td>
<td>GetTrendWindow</td>
</tr>
<tr>
<td>GetToNext</td>
<td>MoveToNext</td>
</tr>
<tr>
<td>GetToNextLine</td>
<td>MoveToNextLine</td>
</tr>
<tr>
<td>GetToPrevious</td>
<td>MoveToPrevious</td>
</tr>
<tr>
<td>GetToPreviousLine</td>
<td>MoveToPreviousLine</td>
</tr>
<tr>
<td>GetOneToOneView</td>
<td>OneToOneView</td>
</tr>
<tr>
<td>PasteRows</td>
<td>PasteRows</td>
</tr>
<tr>
<td>PreviousColumn</td>
<td>PreviousColumn</td>
</tr>
<tr>
<td>ShowInfoText</td>
<td>ShowInfoText</td>
</tr>
<tr>
<td>ShowLockDialog</td>
<td>ShowLockDialog</td>
</tr>
<tr>
<td>ShowLockList</td>
<td>ShowLockList</td>
</tr>
<tr>
<td>ShowLongTermArchiveList</td>
<td>ShowLongTermArchiveList</td>
</tr>
<tr>
<td>ShowMessageList</td>
<td>ShowMessageList</td>
</tr>
<tr>
<td>ShowPercentageAxis</td>
<td>ShowPercentageAxis</td>
</tr>
<tr>
<td>ShowPropertyDialog</td>
<td>ShowPropertyDialog</td>
</tr>
<tr>
<td>ShowSelectArchive</td>
<td>ShowSelectArchive</td>
</tr>
<tr>
<td>ShowSelection</td>
<td>ShowSelection</td>
</tr>
<tr>
<td>ShowSelectionDialog</td>
<td>ShowSelectionDialog</td>
</tr>
<tr>
<td>ShowSelectTimeBase</td>
<td>ShowSelectTimeBase</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>GetTrendWindowCollection</td>
<td>PreviousTrend</td>
</tr>
<tr>
<td>GetValueAxis</td>
<td>Print</td>
</tr>
<tr>
<td>GetValueAxisCollection</td>
<td>QuitTrend</td>
</tr>
<tr>
<td>GetValueColumn</td>
<td>QuitSelected</td>
</tr>
<tr>
<td>GetValueColumnCollection</td>
<td>QuitVisible</td>
</tr>
<tr>
<td>GetXAxis</td>
<td>Read</td>
</tr>
<tr>
<td>GetXAxisCollection</td>
<td>ReadTags</td>
</tr>
<tr>
<td>GetYAxis</td>
<td>Refresh</td>
</tr>
<tr>
<td>GetYAxisCollection</td>
<td>Remove</td>
</tr>
<tr>
<td>HideAlarm</td>
<td>RemoveAll</td>
</tr>
<tr>
<td>Item Method</td>
<td>Restore</td>
</tr>
<tr>
<td>LockAlarm</td>
<td>SelectedStatisticArea</td>
</tr>
<tr>
<td>LoopInAlarm</td>
<td>ServerExport</td>
</tr>
<tr>
<td>MoveAxis</td>
<td>ServerImport</td>
</tr>
<tr>
<td>MoveRuler (Page 772)</td>
<td>ShowColumnSelection</td>
</tr>
<tr>
<td>MoveToFirst</td>
<td>ShowComment</td>
</tr>
<tr>
<td>MoveToFirstLine</td>
<td>ShowDisplayOptionsDialog</td>
</tr>
<tr>
<td>MoveToFirstPage</td>
<td>ShowEmergencyQuitDialog</td>
</tr>
<tr>
<td>MoveToLast</td>
<td>ShowHelp</td>
</tr>
<tr>
<td>MoveToLastLine</td>
<td>ShowHideList</td>
</tr>
<tr>
<td>MoveToLastPage</td>
<td>ShowHitList</td>
</tr>
</tbody>
</table>

**GetRulerBlock** and **GetRulerBlockCollection** are used for manipulating the ruler in the WinCC interface. **GetRulerColumn** and **GetRulerColumnCollection** are used for manipulating the ruler's columns. **MESSAGE** and **MESSAGECOLLECTION** are used for handling messages and message collections. **RULER** and **RULERCOLUMN** are used for manipulating the ruler and its columns.
1.14.5.2 Methods A to E

Activate Method

Function
Activates the specified picture and picture element, respectively.

Note
Focus assignments should not be configured during a ButtonDown event. Since the focus is specifically requested during the ButtonDown event, invalid states may occur.

Syntax
Expression.Activate

Expression
Necessary. An expression which returns an object of type "Screen" or "ScreenItem".

Parameters
- -

Examples
The following example shows the use for type "Screen":

'VBS98
Dim objScreen
MsgBox HMIRuntime.ActiveScreen.ObjectName 'Output of active screen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.Activate 'Activate "ScreenWindow1"
MsgBox HMIRuntime.ActiveScreen.ObjectName 'New output of active screen

The following example shows the use for type "ScreenItem":

'VBS158
MsgBox HMIRuntime.ActiveScreen.ActiveScreenItem.ObjectName 'Output of active screen item
HMIRuntime.ActiveScreen.ScreenItems("IOField1").Activate
MsgBox HMIRuntime.ActiveScreen.ActiveScreenItem.ObjectName 'New output of active screen item
ActivateDynamic method

Function
Dynamically activates a trigger for the defined property and with the defined cycle during runtime. Every time the trigger is activated a different activation cycle can be used.
Examples of this method are available in chapter "VBS for creating procedures and action > Creating and editing actions > Trigger > Animation trigger".

Syntax
Expression.ActivateDynamic (ByVal bstrPropertyName As String, ByVal bstrCycleName As String)

Parameters
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrPropertyName</td>
<td>Name of property to which trigger relates.</td>
</tr>
<tr>
<td>bstrCycleName</td>
<td>Name of activation cycle, e.g. &quot;CycleTime1s&quot;.</td>
</tr>
</tbody>
</table>

See also
Animation trigger (Page 68)

Add Method

Description of TagSet Object
Adds a tag to the list. A tag may be added to the tag object by using name or reference.

Syntax
Expression.Add [Tag]
Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Name of a WinCC tag or reference to a tag object to be added to the list.</td>
</tr>
</tbody>
</table>

Example:

In the following example, a TagSet object is generated and a tag is added.

'VBS170
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"

Tag objects may also be added as follows.

'VBS171
Dim Tag
Set Tag = HMIRuntime.Tags("Motor2")
Dim group2
Set group2 = HMIRuntime.Tags.CreateTagSet
group2.Add Tag

Description of DataSet Object

Adds a value or object reference to the list.

Note

The Data Set Object does not support classes. Objects of type Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet cannot be included in the DataSet list. For object references it must be ascertained that objects are multiread-enabled.

syntax

Expression.Add [vtName], [vtUserData]
Expression
Necessary. An expression which returns an object of type "DataSet".

Parameters
VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vtName</td>
<td>Name by which value or tag are to be added to list.</td>
</tr>
<tr>
<td>vtUserData</td>
<td>Value to be added to list.</td>
</tr>
</tbody>
</table>

Example:
In this example, a value is included in the DataSet list.

'VBS172
HMIRuntime.DataSet.Add "Motor1",23

See also
TagSet Object (List) (Page 141)
DataSet Object (List) (Page 116)

AttachDB method

Function
Executes the “Connect backup” key function of the control.

Syntax
Ausdruck.AttachDB()

Expression
Necessary. An expression that returns an object of the “ScreenItem” type.

Parameters
- -
CalculateStatistic method

Function

Executes the "Calculate statistics" key function of the OnlineTrendControl and OnlineTableControl.

Syntax

Ausdruck.CalculateStatistic()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

---

CopyRows method

Function

Executes the "Copy lines" key function of the control.

Syntax

Ausdruck.CopyRows()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

---

Create method

Function

Creates a new Alarm object.

Syntax

Expression.Create (VARIANT vtApplication)
Expression
Necessary. An expression which returns an object of type "Alarm".

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vtApplication</td>
<td>Name of alarm object (optional)</td>
</tr>
</tbody>
</table>

See also
Alarms object (list) (Page 110)

CreateTagSet Method

Function
Creates a new TagSet object. This object may be used for optimized multi-tag access.

Syntax
Expression.CreateTagSet()

Example:
The following example shows how to create a TagSet object.

'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
See also

TagSet Object (List) (Page 141)
Tags Object (List) (Page 140)

CutRows method

Function

Executes the "Cut lines" key function of the UserArchiveControl.

Syntax

Ausdruck.CutRows()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-

DeactivateDynamic method

Function

Deactivates the trigger used with the "ActivateDynamic" method for the defined property during runtime.

Syntax

Ausdruck.DeactivateDynamic(ByVal bstrPropertyName As String)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

String

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrPropertyName</td>
<td>Name of property to which trigger relates.</td>
</tr>
</tbody>
</table>
DeleteRows method

Function

Executes the "Delete Rows" key function of the UserArchiveControl.

Syntax

Ausdruck.DeleteRows()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

DetachDB method

Function

Executes the "Disconnect backup" key function of the control.

Syntax

Ausdruck.DetachDB()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Edit method

Function

Executes the "Edit" key function of the OnlineTableControl.

Syntax

Ausdruck.Edit()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
-

Export Method

Function
Executes the "Export archive" or "Export data" key function of the control.

Syntax
Ausdruck.Export()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
VARIANT

1.14.5.3 Get methods

GetColumn method

Function
Returns the name or index designated column object of the WinCC UserArchiveControl as type "ICCAxUAColumn".

Syntax
Ausdruck.GetColumn(ByVal vIndex As Variant)

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of column of UserArchiveControl.</td>
</tr>
</tbody>
</table>

Example

'VBS312
Dim ctrl
Dim objColumn
Set ctrl = ScreenItems("UAControl")
Set objColumn = ctrl.GetColumn("Field1")
objColumn.Length = 30
Set objColumn = ctrl.GetColumn(3)
objColumn.Align = 2

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "Column" listing, for example, you write "objColumn.Align" instead of "objColumn.ColumnAlign".

See also

Column object (list) (Page 221)

GetColumnCollection method

Function

Returns the list of all column objects of the WinCC UserArchiveControl as type "ICCAxCollection".

Syntax

Ausdruck.GetColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS313
Dim ctrl
Dim coll
Dim field
Set ctrl = ScreenItems("UAControl")
Set coll = ctrl.GetColumnCollection
HMIRuntime.Trace "Number of fields:" & coll.Count & vbCrLf
For Each field In coll
    HMIRuntime.Trace field.Name & vbCrLf
    HMIRuntime.Trace field.Type & vbCrLf
    HMIRuntime.Trace field.Length & vbCrLf
    HMIRuntime.Trace field.Caption & vbCrLf
Next

See also

Column object (list) (Page 221)

GetHitlistColumn method

Function

Returns the name or index designated column object of the hitlist of the WinCC AlarmControl as type "ICCAxMessageColumn".

Syntax

Expression.GetHitlistColumn(ByVal vIndex As Variant)
Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of hitlist column</td>
</tr>
</tbody>
</table>

Example

```
'VBS314
Dim ctrl
Dim objHitlistColumn
Set ctrl = ScreenItems("AlarmControl")
Set objHitlistColumn = ctrl.GetHitlistColumn("Date")
objHitlistColumn.Sort = 2
Set objHitlistColumn = ctrl.GetHitlistColumn("AverageComeGo")
objHitlistColumn.Visible = FALSE
```

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "HitlistColumn" listing, for example, you write "objHitlistColumn.Visible" instead of "objHitlistColumn.HitlistColumnVisible".

See also

HitlistColumn object (list) (Page 222)

GetHistlistColumnCollection method

Function

Returns the list of all column objects of the WinCC AlarmControl hitlist as type "ICCAxCollection".

Syntax

```
Expression.GetHitlisteColumnCollection()
```
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
-

Features and functions of the ICCAxCollection
The following properties are available for the ICCAxCollection:
- Count
- Item

The following functions are available for the ICCAxCollection:
- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS315
Dim ctrl
Dim coll
Dim hitlistcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetHitlistColumnCollection
HMIRuntime.Trace "Number of hitlist columns:" & coll.Count & vbCrLf
For Each hitlistcol In coll
    HMIRuntime.Trace hitlistcol.Index & vbCrLf
    HMIRuntime.Trace hitlistcol.Name & vbCrLf
    HMIRuntime.Trace hitlistcol.Sort & vbCrLf
    HMIRuntime.Trace hitlistcol.SortIndex & vbCrLf
Next
```

See also
HitlistColumn object (list) (Page 222)

GetMessageBlock method

Function
Returns the name or index designated message block object of the WinCC AlarmControl as type "ICCAxMessageBlock".
Syntax

`Expression.GetMessageBlock(ByVal vIndex As Variant)`

**Expression**

Necessary. An expression that returns an object of the “ScreenItem” type.

**Parameters**

**VARIANT**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of message block.</td>
</tr>
</tbody>
</table>

**Example**

```vbs
'VBS316
Dim ctrl
Dim objMsgBlock
Set ctrl = ScreenItems("AlarmControl")
Set objMsgBlock = ctrl.GetMessageBlock("Date")
objMsgBlock.Align = 2
Set objMsgBlock = ctrl.GetMessageBlock("Number")
objMsgBlock.LeadingZeros = 4
```

**Note**

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "MessageBlock" listing, for example, you write "objMsgBlock.Align" instead of "objMsgBlock.MessageBlockAlign".

**See also**

MessageBlock object (list) (Page 223)

**GetMessageBlockCollection method**

**Function**

Returns the list of all message block objects of the WinCC AlarmControl as type "ICCAxCollection".
Syntax

Expression.GetMessageBlockCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:
- Count
- Item

The following functions are available for the ICCAxCollection:
- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS317
Dim ctrl
Dim coll
Dim msgblock
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageBlockCollection
For Each msgblock In coll
    msgblock.Align = 1
    msgblock.Length = 12
    msgblock.Selected = TRUE
Next

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "MessageBlock" listing, for example, you write "msgblock.Align" instead of "msgblock.MessageBlockAlign".

See also

MessageBlock object (list) (Page 223)
**GetMessageColumn method**

**Function**

Returns the name or index designated column object of the WinCC AlarmControl as type "ICCAxMessageColumn".

**Syntax**

```
Expression.GetMessageColumn(ByVal vIndex As Variant)
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of column in message list.</td>
</tr>
</tbody>
</table>

**Example**

```vbs
'VBS318
Dim ctrl
Dim objMessColumn
Set ctrl = ScreenItems("AlarmControl")
Set objMessColumn = ctrl.GetMessageColumn("Date")
objMessColumn.Visible = FALSE
Set objMessColumn = ctrl.GetMessageColumn("Number")
objMessColumn.Sort = 1
```

**Note**

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "MessageColumn" listing, for example, you write "objMessColumn.Visible" instead of "objMessColumn.MessageColumnVisible".

**See also**

MessageColumn object (list) (Page 223)
GetMessageColumnCollection method

Function

Returns the list of all column objects of the WinCC AlarmControl as type "ICCAxCollection".

Syntax

Expression.GetMessageColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS319
Dim ctrl
Dim coll
Dim msgcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageColumnCollection
HMIRuntime.Trace "Number of message columns:" & coll.Count & vbCrLf
For Each msgcol In coll
    HMIRuntime.Trace msgcol.Index & vbCrLf
    HMIRuntime.Trace msgcol.Name & vbCrLf
    HMIRuntime.Trace msgcol.Sort & vbCrLf
    HMIRuntime.Trace msgcol.SortIndex & vbCrLf
Next
See also

MessageColumn object (list) (Page 223)

GetOperatorMessage method

Function

Returns the name or index designated operator message object of the WinCC AlarmControl as type "ICCAxOperatorMessage".

Syntax

Expression.GetOperatorMessage(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of operator message</td>
</tr>
</tbody>
</table>

Example

```vbs
'VBS320
Dim ctrl
Dim objOpMess
Set ctrl = ScreenItems("AlarmControl")
Set objOpMess = ctrl.GetOperatorMessage(0)
objOpMess.Source1 = "Number"
objOpMessSourceType1 = 1
```

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "OperatorMessage" listing, for example, you write "objOpMess.Source1" instead of "objOpMess.OperatorMessageSource1".
See also

OperatorMessage object (list) (Page 224)

GetOperatorMessageCollection method

Function

Returns the list of all operator message objects of the WinCC AlarmControl as type "ICCAxCollection".

Syntax

Expression.GetOperatorMessageCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS321
Dim ctrl
Dim coll
Dim opmsg
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetOperatorMessageCollection
For Each opmsg In coll
    HMIRuntime.Trace opmsg.Index & vbCrLf
    HMIRuntime.Trace opmsg.Name & vbCrLf
    HMIRuntime.Trace opmsg.Number & vbCrLf
    HMIRuntime.Trace opmsg.Selected & vbCrLf
Next
See also

OperatorMessage object (list) (Page 224)

GetRow method

Function

Returns the row number designated row object of the table-based controls as type "ICCAxDataRow".

Syntax

Expression.GetRow(ByVal IRow As Long)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Long

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRow</td>
<td>Number of the desired line of the control.</td>
</tr>
</tbody>
</table>

Example

'VBS356
Dim ctrl
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("UAControl")
Set coll = ctrl.GetRowCollection
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
    HMIRuntime.trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & "  
'enumerate and trace out column titles and cell texts
For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
    HMIRuntime.Trace ctrl.GetRow(0).CellText(lCellIndex) & "  
    HMIRuntime.trace ctrl.GetRow(lIndex).CellText(lCellIndex) & "  
Next
HMIRuntime.trace vbNewLine
Next
Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also
Row object (list) (Page 225)

GetRowCollection method

Function
Returns the list of all row objects of the table-based controls type "ICCAxDataRowCollection".

Syntax
Expression.GetRowCollection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

Properties of the ICCAxDataRowCollection
The ICCAxDataRowCollection refers to runtime data. The data is read-only. It is not possible to add and edit the data.
The following properties are available for the ICCAxDataRowCollection:

- Count - Determines the number of rows in the collection.
- Item - Access to an individual row within the collection via the row number. Numbering runs from 1 to Count. A Row object is returned.
Example

'VBS357
Dim ctrl
Dim coll
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetRowCollection
HMIRuntime.Trace "Number of message rows:" & coll.Count & vbCrLf
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
    HMIRuntime.Trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & vbCrLf
'enumerate and trace out column titles and cell texts
For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
    HMIRuntime.Trace ctrl.GetMessageColumn(lCellIndex -1).Name & " "
    HMIRuntime.Trace ctrl.GetRow(lIndex).CellText(lCellIndex) & " "
Next
HMIRuntime.Trace vbNewLine
Next

See also
Row object (list) (Page 225)

GetRulerBlock method

Function

Returns the Block object designated as name or index of the WinCC RulerControl as type "ICCAxRulerBlock".

Syntax

Expression.GetRulerBlock(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of block in RulerControl</td>
</tr>
</tbody>
</table>
Example

'VBS322
Dim ctrl
Dim objRulerBlock
Set ctrl = ScreenItems("RulerControl")
Set objRulerBlock = ctrl.GetRulerBlock(0)
objRulerBlock.Caption = "RulerBlock1"
Set objRulerBlock = ctrl.GetRulerBlock("Name")
objRulerBlock.Length = 10

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "RulerBlock" listing, for example, you write "objRulerBlock.Caption" instead of "objRulerBlock.BlockCaption".

See also
RulerBlock object (list) (Page 226)

GetRulerBlockCollection method

Function
Returns the list of all Block objects of the WinCC RulerControl as type "ICCAxCollection".

Syntax
Expression.GetRulerBlockCollection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

Features and functions of the ICCAxCollection
The following properties are available for the ICCAxCollection:

- Count
- Item
The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS323
Dim ctrl
Dim coll
Dim rulerblock
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerBlockCollection
For Each rulerblock In coll
   rulerblock.Align = 1
   rulerblock.Length = 12
Next

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "RulerBlock" listing, for example, you write "rulerblock.Align" instead of "rulerblock.RulerBlockAlign".

See also
RulerBlock object (list) (Page 226)

GetRulerColumn method

Function
Returns the Column object designated as name or index of the WinCC RulerControl as type "ICCAxRulerColumn".

Syntax
Expression.GetRulerColumn(ByVal vIndex As Variant)

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of column of RulerControl.</td>
</tr>
</tbody>
</table>

Example

'VBS324
Dim ctrl
Dim objRulercol
Set ctrl = ScreenItems("RulerControl")
Set objRulercol = ctrl.GetRulerColumn("Name")
objRulercol.Sort = 0
Set objRulercol = ctrl.GetRulerColumn("ValueY")
objRulercol.Visible = FALSE

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "RulerColumn" listing, for example, you write "objRulercol.Visible" instead of "objRulercol.ColumnVisible".

See also

RulerColumn object (list) (Page 227)

GetRulerColumnCollection method

Function

Returns the list of all Column objects of the WinCC RulerControl as type "ICCAxCollection".

Syntax

Expression.GetRulerColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

● Count
● Item

The following functions are available for the ICCAxCollection:

● AddItem(vName) As Object
● RemoveItem(vIndex)

Example

'VBS325
Dim ctrl
Dim coll
Dim rulercol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerColumnCollection
HMIRuntime.Trace "Number of ruler columns:" & coll.Count & vbCrLf
For Each rulercol In coll
  HMIRuntime.Trace rulercol.Index & vbCrLf
  HMIRuntime.Trace rulercol.Name & vbCrLf
  HMIRuntime.Trace rulercol.Sort & vbCrLf
  HMIRuntime.Trace rulercol.SortIndex & vbCrLf
Next

See also

RulerColumn object (list) (Page 227)

GetRulerData method

Function

Returns the value of the called trend at the ruler position.

Syntax

Expression.GetRulerData(ByVal RulerIndex As Long, pvValue As Variant, Optional pvTimeStamp As Variant, Optional pvFlags As Variant) Long

Expression

Necessary. An expression which returns an object of the "Trend" type.
Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RulerIndex</td>
<td>0 = Ruler</td>
</tr>
<tr>
<td>pvValue</td>
<td>Value of X axis</td>
</tr>
<tr>
<td>pvTimeStamp</td>
<td>Time or value of the Y axis</td>
</tr>
<tr>
<td>pvFlags</td>
<td>Quality code</td>
</tr>
</tbody>
</table>

Example

'VBS326
Dim ctrl
Dim objTrend
Dim objIOField1
Dim objIOField2
    Dim value
Dim time
Set ctrl = ScreenItems("Control1")
Set objTrend = ctrl.GetTrend("Trend 1")
Set objIOField1 = ScreenItems("I/O Field1")
Set objIOField2 = ScreenItems("I/O Field2")
objTrend.GetRulerData 0, value, time
objIOField1.OutputValue = value
objIOField2.OutputValue = time

GetSelectedRow method

Function

Returns the selected row object of the table-based controls as type "ICCAxDataRow".

Syntax

Expression.GetSelectedRow()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -
Example

'VBS358
Dim ctrl
Dim lCellIndex
Dim lCellCount
Dim headingRow
Dim selectedRow
Set ctrl = ScreenItems("TableControl")
Set headingRow = ctrl.GetRow(0)
Set selectedRow = ctrl.GetSelectedRow
lCellCount = headingRow.CellCount
'enumerate and trace out column titles and cell texts
For lCellIndex = 1 To lCellCount
    HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
    HMIRuntime.trace selectedRow.CellText(lCellIndex)
    HMIRuntime.trace vbNewLine
Next

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also
Row object (list) (Page 225)

GetSelectedRows method

Function
Returns the selected row objects of the table-based controls as type "ICCAxDataRow" for multiple selection.

Syntax
Expression.GetSelectedRows()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

Example

'VBS359
Dim ctrl
Dim lCellIndex
Dim lCellCount
Dim lRowIndex
Dim lRowCount
Dim headingRow
Dim selectedRow
Dim selectedRows
Set ctrl = ScreenItems("TableControl")
Set headingRow = ctrl.GetRow(0)
Set selectedRows = ctrl.GetSelectedRows
lCellCount = headingRow.CellCount
lRowCount = selectedRows.Count
'enumerate selected rows
For lRowIndex = 1 To lRowCount
    Set selectedRow = selectedRows(lRowIndex)
    HMIRuntime.Trace "Row number: " & CStr(lRowIndex) & vbNewLine
'enumerate and trace out column titles and cell texts
For lCellIndex = 1 To lCellCount
    HMIRuntime.trace headingRow.CellText(lCellIndex) & " : "
    HMIRuntime.trace selectedRow.CellText(lCellIndex)
    HMIRuntime.trace vbNewLine
Next
Next

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

Row object (list) (Page 225)
GetStatisticAreaColumn method

Function

Returns the name or index designated Column object of the WinCC RulerControl statistics area window as type "ICCAxRulerColumn".

Syntax

Ausdruck.GetStatisticAreaColumn(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of column of statistics area window.</td>
</tr>
</tbody>
</table>

Example

'VBS327
Dim ctrl
Dim objStatAreaCol
Set ctrl = ScreenItems("RulerControl")
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("DatasourceY")
objStatAreaCol.Visible = FALSE
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("ValueY(LL)")
objStatAreaCol.Sort = 1

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "StatisticAreaColumn" listing, for example, you write "objStatAreaCol.Visible" instead of "objStatAreaCol.ColumnVisible".

See also

StatisticAreaColumn object (list) (Page 228)
GetStatisticAreaColumnCollection method

Function

Returns the list of all column objects of the WinCC RulerControl statistics area window as type "ICCAxCollection".

Syntax

Ausdruck.GetStatisticAreaColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS328
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticAreaColumnCollection
HMIRuntime.Trace "Number of statistic Area columns:" & coll.Count & vbCrLf
For Each statcol In coll
    HMIRuntime.Trace statcol.Index & vbCrLf
    HMIRuntime.Trace statcol.Name & vbCrLf
    HMIRuntime.Trace statcol.Sort & vbCrLf
    HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
GetStatisticResultColumn method

Function

Returns the name or index designated Column object of the WinCC RulerControl statistics window as type "ICCAxRulerColumn".

Syntax

Ausdruck.GetStatisticResultColumn(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of column of statistics window.</td>
</tr>
</tbody>
</table>

Example

'VBS329
Dim ctrl
Dim objStatResCol
Set ctrl = ScreenItems("RulerControl")
Set objStatResCol = ctrl.GetStatisticResultColumn("MaxValue")
ojStatResCol.Visible = FALSE
Set objStatResCol = ctrl.GetStatisticResultColumn("Average")
ojStatResCol.Sort = 2

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "StatisticResultColumn" listing, for example, you write "objStatResCol.Visible" instead of "objStatResCol.ColumnVisible".

See also

StatisticResultColumn object (list) (Page 229)
GetStatisticResultColumnCollection method

Function

Returns the list of all Column objects of the WinCC RulerControl statistics window as type "ICCAxCollection".

Syntax

Ausdruck.GetStatisticResultColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS330
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticResultColumnCollection
HMIRuntime.Trace "Number of statistic result columns:" & coll.Count & vbCrLf
For Each statcol In coll
    HMIRuntime.Trace statcol.Index & vbCrLf
    HMIRuntime.Trace statcol.Name & vbCrLf
    HMIRuntime.Trace statcol.Sort & vbCrLf
    HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
See also

StatisticResultColumn object (list) (Page 229)

GetStatusBarElement method

Function

Returns the element of the control status bar designated as name or index as type "ICCAxStatusBarElement".

Syntax

Ausdruck.GetStatusBarElement(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of status bar element.</td>
</tr>
</tbody>
</table>

Example

'VBS331
Dim ctrl
Dim objStatusBar
Set ctrl = ScreenItems( "Control1" )
Set objStatusBar = ctrl.GetStatusBarElement(1)
objStatusBar.Visible = FALSE
Set objStatusBar = ctrl.GetStatusBarElement(3)
objStatusBar.Width = 10

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "StatusBarElement" listing, for example, you write "objStatusBar.Visible" instead of "objStatusBar.StatusbarElement.Visible".
GetStatusBarElementCollection method

Function

Returns the list of all status bar elements of the control as type "ICCAxCollection".

Syntax

\[ \text{Ausdruck.GetStatusBarElementCollection()} \]

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS332
Dim ctrl
Dim coll
Dim statelement
Set ctrl = ScreenItems.Item("Control1")
Set coll = ctrl.GetStatusBarElementCollection
HMIRuntime.Trace "Number of statusbar elements:" & coll.Count & vbCrLf
For Each statelement In coll
    HMIRuntime.Trace statelement.Name & vbCrLf
    HMIRuntime.Trace statelement.Width & vbCrLf
    HMIRuntime.Trace statelement.Text & vbCrLf
Next
Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "StatusbarElement" listing, for example, you write "statelement.Name" instead of "statelement.StatusbarElementName".

See also

StatusbarElement object (list) (Page 229)

GetTimeAxis method

Function

Returns the time axis object designated as name or index of the WinCC OnlineTrendControl as type "ICCAxTimeAxis".

Syntax

Ausdruck.GetTimeAxis(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of time axis.</td>
</tr>
</tbody>
</table>
Example

'VBS333
Dim ctrl
Dim objTimeAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTimeAxis = ctrl.GetTimeAxis(1)
objTimeAxis.Visible = FALSE
Set objTimeAxis = ctrl.GetTimeAxis("axis 2")
objTimeAxis.Label = "Time axis 2"
objTimeAxis.DateFormat = "dd.MM.yy"
objTimeAxis.TimeFormat = "HH:mm:ss.ms"
objTimeAxis.RangeType = 2
objTimeAxis.BeginTime = "06.04.2010 9:33:18"
objTimeAxis.MeasurePoints = 100

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "TimeAxis" listing, for example, you write "objTimeAx.Visible" instead of "objTimeAx.TimeAxisVisible".

See also
TimeAxis object (list) (Page 230)

GetTimeAxisCollection method

Function
Returns the list of all time axis objects of the WinCC OnlineTrendControl as type "ICCAxCollection".

Syntax
Ausdruck.GetTimeAxisCollection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS334
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis1
Dim objTimeAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis1 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2010")
Set objTimeAxis2 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2011")
objTimeAxis1.TrendWindow = objTrendWnd.Name
objTimeAxis1.Label = "2010"
objTimeAxis1.RangeType = 1
objTimeAxis1.BeginTime = "01.01.2010 0:00:00"
objTimeAxis2.TrendWindow = objTrendWnd.Name
objTimeAxis2.Label = "2011"
objTimeAxis2.RangeType = 1
objTimeAxis2.BeginTime = "01.01.2011 0:00:00"
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis2.Name

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "TimeAxis" listing, for example, you write "objTimeAxis1.Label" instead of "objTimeAxis1.TimeAxisLabel".
See also

TimeAxis object (list) (Page 230)

GetTimeColumn method

Function

Returns the time column object designated as name or index of the WinCC OnlineTableControl as type "ICCAxTimeColumn".

Syntax

Ausdruck.GetTimeColumn(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of time column.</td>
</tr>
</tbody>
</table>

Example

'VBS335
Dim ctrl
Dim objTimeCol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn1")
objTimeCol.ShowDate = FALSE
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn2")
objTimeCol.Visible = FALSE

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "TimeColumn" listing, for example, you write "objTimeColumn.ShowDate" instead of "objTimeColumn.TimeColumnShowDate".

See also

TimeColumn object (list) (Page 231)
GetTimeColumnCollection method

Function

Returns the list of all time column objects of the WinCC OnlineTableControl as type "ICCAxCollection".

Syntax

Ausdruck.GetTimeColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)
Example

'VBS336
Dim ctrl
Dim objTimeCol1
Dim objTimeCol2
Dim coll
Dim timecol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol1 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2010")
Set objTimeCol2 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2011")
objTimeCol1.Caption = "2010"
objTimeCol1.RangeType = 1
objTimeCol1.BeginTime = "01.01.2010 0:00:00"
objTimeCol2.Caption = "2011"
objTimeCol2.RangeType = 0
objTimeCol2.BeginTime = "01.01.2011 0:00:00"
objTimeCol2.TimeRangeFactor = 1
objTimeCol2.TimeRangeBase = 3600000
Set coll = ctrl.GetTimeColumnCollection
For Each timecol In coll
    timecol.Align = 1
    timecol.Length = 12
    timecol.BackColor = RGB(240,240,0)
    timecol.ForeColor = RGB(130,160,255)
Next

See also

TimeColumn object (list) (Page 231)

GetToolbarButton method

Function

Returns the name or index designated toolbar button function of the control as type "ICCAxToolbarButton".

Syntax

Ausdruck.GetToolbarButton(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of toolbar button function.</td>
</tr>
</tbody>
</table>

Example

'VBS337
Dim ctrl
Set ctrl = ScreenItems( "Controll" )
Dim toolbu
Set toolbu = ctrl.GetObjectButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "ToolbarButton" listing, for example, you write "toolbu.Index" instead of "toolbu.ToolbarButtonIndex".

See also

ToolbarButton object (list) (Page 232)

GetToolbarButtonCollection method

Function

Returns the list of all toolbar button functions of the control as type "ICCAxCollection".

Syntax

Ausdruck.GetToolbarButtonCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -
Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following methods are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS338
Dim ctrl
Dim coll
Dim toolbu
Set ctrl = ScreenItems( "Control1" )
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
For Each toolbu In coll
    HMIRuntime.Trace toolbu.Name & vbCrLf
    HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
    HMIRuntime.Trace "Authorization: " & toolbu.PasswordLevel & vbCrLf
Next

See also

ToolbarButton object (list) (Page 232)

GetTrend method

Function

Returns the trend object designated as name or index of the WinCC OnlineTrendControl or WinCC FunctionTrendControl as type "ICCAxTrend" or "ICCAxFunctionTrend".

Syntax

Ausdruck.GetTrend(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of curve.</td>
</tr>
</tbody>
</table>

Example

`'VBS339
Dim ctrl
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrend = ctrl.GetTrend("Trend 1")
objTrend.PointStyle = 1
objTrend.LineWidth = 4
Set objTrend = ctrl.GetTrend(2)
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag2"
``

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "Trend" listing, for example, you write "objTrend.PointStyle" instead of "objTrend.TrendPointStyle".

See also

Trend object (list) (Page 233)

GetTrendCollection method

Function

Returns the list of all trend objects of the WinCC OnlineTrendControl or WinCC FunctionTrendControl as type "ICCAxCollection".

Syntax

`Ausdruck.GetTrendCollection()`

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS340
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag1"
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValAxis.Name

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "Trend" listing, for example, you write "objTrend.TagName" instead of "objTrend.TrendTagName".

See also

Trend object (list) (Page 233)
GetTrendWindow method

Function
Returns the trend window object designated as name or index of the WinCC OnlineTrendControl or WinCC FunctionTrendControl as type "ICCAxTrendWindow".

Syntax
Ausdruck.GetTrendWindow(ByVal vIndex As Variant)

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of curve window.</td>
</tr>
</tbody>
</table>

Example

'VBS341
Dim ctrl
Dim objTrendWnd
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindow(1)
objTrendWnd.Visible = FALSE
Set objTrendWnd = ctrl.GetTrendWindow("trend window 2")
objTrendWnd.VerticalGrid = TRUE
objTrendWnd.FineGrid = TRUE

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "TrendWindow" listing, for example, you write "objTrendWnd.Visible" instead of "objTrendWnd.TrendWindowVisible".

See also
TrendWindow object (list) (Page 234)
GetTrendWindowCollection method

Function

Returns the list of all trend window objects of the WinCC OnlineTrendControl or WinCC FunctionTrendControl as type "ICCAxCollection".

Syntax

Ausdruck.GetTrendWindowCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

---

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS342
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name

See also

TrendWindow object (list) (Page 234)
GetValueAxis method

Function

Returns the value axis object designated as name or index of the WinCC OnlineTrendControl as type "ICCAxValueAxis".

Syntax

Ausdruck.GetValueAxis(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of value axis.</td>
</tr>
</tbody>
</table>

Example

'VBS343
Dim ctrl
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objValAxis = ctrl.GetValueAxis(1)
objValAxis.Visible = FALSE
Set objValAxis = ctrl.GetValueAxis("axis 2")
objValAxis.Label = "Value axis 2"
objValAxis.ScalingType = 0
objValAxis.Precisions = 2
objValAxis.AutoRange = TRUE

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "ValueAxis" listing, for example, you write "objValueAx.Visible" instead of "objValueAx.ValueAxisVisible".

See also

ValueAxis object (list) (Page 235)
GetValueAxisCollection method

Function

Returns the list of all value axis objects of the WinCC OnlineTrendControl as type "ICCAxCollection".

Syntax

Ausdruck.GetValueAxisCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)
Example

'VBS344
Dim ctrl
Dim objTrendWnd
Dim objValAxis1
Dim objValAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objValAxis1 = ctrl.GetValueAxisCollection.AddItem("myValueAxis1")
Set objValAxis2 = ctrl.GetValueAxisCollection.AddItem("myValueAxis2")
objValAxis1.TrendWindow = objTrendWnd.Name
objValAxis1.Label = "Value1"
objValAxis2.TrendWindow = objTrendWnd.Name
objValAxis2.inTrendColor = TRUE
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis2.Name

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "ValueAxis" listing, for example, you write "objValueAxis1.Label" instead of "objValueAxis1.ValueAxisLabel".

See also
ValueAxis object (list) (Page 235)

GetValueColumn method

Function
Returns the column object designated as name or index of the WinCC OnlineTableControl as type "ICCAxValueColumn".

Syntax
Ausdruck.GetValueColumn(ByVal vIndex As Variant)

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of value column of OnlineTable-Control.</td>
</tr>
</tbody>
</table>

Example

```vbs
'VBS345
Dim ctrl
Dim objValueColumn
Set ctrl = ScreenItems("TableControl")
Set objValueColumn = ctrl.GetValueColumn("Valuecolumn1")
objValueColumn.Precisions = 4
Set objValueColumn = ctrl.GetValueColumn(2)
objValueColumn.ExponentialFormat = TRUE
```

**Note**

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "ValueColumn" listing, for example, you write "objValueColumn.Precisions" instead of "objValueColumn.ValueColumnPrecisions".

**See also**

ValueColumn object (list) (Page 236)

**GetValueColumnCollection method**

**Function**

Returns the list of all value column objects of the WinCC OnlineTableControl as type "ICCAxCollection".

**Syntax**

```vbs
Ausdruck.GetValueColumnCollection()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS346
Dim ctrl
Dim objValCol1
Dim objValCol2
Dim coll
Dim valcol
Set ctrl = ScreenItems("TableControl")
Set objValCol1 = ctrl.GetValueColumnCollection.AddItem("ValueColumn1")
Set objValCol2 = ctrl.GetValueColumnCollection.AddItem("ValueColumn2")
objValCol1.Caption = "Value Archive"
objValCol1.Provider = 1
objValCol1.TagName = "ProcessValueArchive\arch1"
objValCol1.TimeColumn = "TimeColumn1"
objValCol1.Caption = "Value Tag"
objValCol2.Provider = 2
objValCol2.TagName = "tagxx"
objValCol2.TimeColumn = "TimeColumn2"
Set coll = ctrl.GetValueColumnCollection
For Each valcol In coll
    valcol.Align = 2
    valcol.Length = 10
    valcol.AutoPrecisions = TRUE
Next

See also

ValueColumn object (list) (Page 236)
GetXAxis method

Function

Returns the X axis object designated as name or index of the WinCC FunctionTrendControl as type "ICCAxValueAxis".

Syntax

Ausdruck.GetXAxis(ByVal vIndex As Variant)

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of X axis.</td>
</tr>
</tbody>
</table>

Example

'VBS347
Dim ctrl
Dim objXAx

Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAx = ctrl.GetXAxis(0)
objXAx.ScalingType = 0
Set objXAx = ctrl.GetXAxis("X axis 1")
objXAx.Label = "X2"
objXAx.Precisions = 2
objXAx.Color = RGB(109,109,109)

Note

If you access the properties with the listing object, you do not have to enter the name of the listing.

For the "XAxis" listing, for example, you write "objXAx.Visible" instead of "objXAx.XAxisVisible".

See also

XAxis object (list) (Page 237)
GetXAxisCollection method

Function

Returns the list of all X axis objects of the WinCC FunctionTrendControl as type "ICCAxCollection".

Syntax

Ausdruck.GetXAxisCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS348
Dim ctrl
Dim objXAxis1
Dim objXAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAxis1 = ctrl.GetXAxisCollection.AddItem("myXAxis1")
objXAxis1.Label = "temperature"
Set objXAxis2 = ctrl.GetXAxisCollection.AddItem("myXAxis2")
objXAxis2.Label = "pressure"
Set coll = ctrl.GetXAxisCollection
HMIRuntime.Trace "Number of XAxis:" & coll.Count & vbCrLf
For Each axes In coll
    HMIRuntime.Trace axes.Name & vbCrLf
    HMIRuntime.Trace axes.Label & vbCrLf
Next
Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "XAxis" listing, for example, you write "objXAxis1.Label" instead of "objXAxis1.XAxisLabel".

See also
XAxis object (list) (Page 237)

GetYAxis method

Function

Returns the Y axis object designated as name or index of the WinCC FunctionTrendControl as type "ICCAxValueAxis".

Syntax

Ausdruck.GetYAxis(ByVal vIndex As Variant)

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vIndex</td>
<td>Index or name of Y axis.</td>
</tr>
</tbody>
</table>
Example

'VBS349
Dim ctrl
Dim objYAx

Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAx = ctrl.GetYAxis(0)
objYAx.Align = 0
objYAx.Precisions = 3

Set objYAx = ctrl.GetYAxis("Y axis 1")
objYAx.Label = "Y1"
Msgbox objYAx.Label
objYAx.EndValue = 90
objYAx.BeginValue = 10

Note
If you access the properties with the listing object, you do not have to enter the name of the listing.
For the "YAxis" listing, for example, you write "objYAx.Visible" instead of "objYAx.YAxisVisible".

See also
YAxis object (list) (Page 238)

GetYAxisCollection method

Function
Returns the list of all Y axis objects of the WinCC FunctionTrendControl of type "ICCAXCollection".

Syntax
Ausdruck.GetYAxisCollection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

'VBS350
Dim ctrl
Dim objYAxis1
Dim objYAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAxis1 = ctrl.GetXAxisCollection.AddItem("myYAxis1")
objYAxis1.Label = "temperature"
Set objYAxis2 = ctrl.GetXAxisCollection.AddItem("myYAxis2")
objYAxis2.Label = "pressure"
Set coll = ctrl.GetYAxisCollection
HMIRuntime.Trace "Number of YAxis:" & coll.Count & vbCrLf
For Each axes In coll
    HMIRuntime.Trace axes.Name & vbCrLf
    HMIRuntime.Trace axes.Label & vbCrLf
Next

Note

If you access the properties with the listing object, you do not have to enter the name of the
listing.

For the "YAxis" listing, for example, you write "objYAxis1.Label" instead of "objYAxis1.YAxisLabel".

See also

YAxis object (list) (Page 238)
1.14.5.4 Methods H to M

HideAlarm method

Function

Executes the "Hide messages" key function of the AlarmControl.

Syntax

Expression.HideAlarm()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-

InsertData method

Function

Adds data to the called trend.

Syntax

Expression.InsertData(dblAxisX As Variant, dblAxisY As Variant)

Expression

Necessary. An expression which returns an object of the "Trend" type.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblAxisX</td>
<td>Value of X axis</td>
</tr>
<tr>
<td>dblAxisY</td>
<td>Value of Y axis</td>
</tr>
</tbody>
</table>
Example

'VBS300
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
For lngFactor = -100 To 100
    dblAxisX = CDb1(lngFactor * 0.02)
    dblAxisY = CDb1(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
    objTrend.InsertData dblAxisX, dblAxisY
Next

Item Method

Function

Retrieves an object from a collection and enables access to it via Index.

Description of DataItem Object

Access uses the name under which the value was added to the list. Single access using an index is not recommended since the index changes during adding or deleting of values.

syntax

Expression.Item()

Expression

Necessary. An expression which returns an object of the type "Screens", "Layers" (or "Tags").

Note

In the case of "Tags", restricted functional scope! The standard methods get_Count and get_NewEnum are missing so that access via Index nor the counting of all tags is possible.

Parameters

VARIANT
Example:

The following example issues the names of all objects contained in the picture "NewPDL1":

```vbs
'VBS99
Dim objScreen
Dim objScrItem
Dim lngIndex
Dim lngAnswer
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
    strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
    Set objScrItem = objScreen.ScreenItems(strName)
    lngAnswer = MsgBox(objScrItem.ObjectName, vbOKCancel)
    If vbCancel = lngAnswer Then Exit For
Next
```

See also

- ScreenItems Object (List) (Page 129)
- ScreenItem Object (Page 125)
- Tags Object (List) (Page 140)
- Alarms object (list) (Page 110)
- ProcessValues Object (List) (Page 124)

LockAlarm method

Function

Executes the "Lock Alarm" key function of the AlarmControl.

Syntax

`Expression.LockAlarm()`

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -
LoopInAlarm method

Function
Executes the "Loop in Alarm" key function of the AlarmControl.

Syntax
Expression.LoopInAlarm()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveAxis method

Function
Executes the "Move axis" key function of the OnlineTrendControl and FunctionTrendControl.

Syntax
Expression.MoveAxis()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveRuler

Function
Moves the ruler from a specified reference point by a specified distance.

Syntax
Expression.MoveRuler( RulerIndex As Long, RulerMoveRef As Long,
MoveDistance As Long, Optional vTrendWindow As Variant )
Expression
Required. An expression that returns an object of the "ScreenItem" type.

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RulerIndex</td>
<td>Specifies the ruler to move:</td>
</tr>
<tr>
<td></td>
<td>0 = Ruler</td>
</tr>
<tr>
<td></td>
<td>1 = Ruler at the start of the statistics area</td>
</tr>
<tr>
<td></td>
<td>2 = Ruler at the end of the statistics area</td>
</tr>
<tr>
<td>RulerMoveRef</td>
<td>Specifies the reference point as orientation for the third parameter &quot;MoveDistance&quot;:</td>
</tr>
<tr>
<td></td>
<td>0 = Time axis start position</td>
</tr>
<tr>
<td></td>
<td>1 = Current ruler position</td>
</tr>
<tr>
<td></td>
<td>2 = Time axis end position</td>
</tr>
<tr>
<td>MoveDistance</td>
<td>Number of pixels by which the ruler is moved away from reference point &quot;RulerMoveRef&quot;.</td>
</tr>
<tr>
<td>vTrendWindow</td>
<td>Optional parameter for handling several, independent trend windows.</td>
</tr>
<tr>
<td></td>
<td>Specifies the trend window in which the ruler is moved. The ruler moves in all trend windows if this parameter is not specified.</td>
</tr>
</tbody>
</table>

Return value
Function that returns the new ruler position.

Example

Table 1-1  Move ruler left by 10 pixels

'VBS367
Sub OnClick(ByVal Item)
  Dim ctrl
  Set ctrl = ScreenItems.Item("Control1")
  call ctrl.MoveRuler (0, 1, -10)
End Sub

In the example, the ruler is moved by -10 pixels, starting at reference point 1 (current ruler position). The ruler is now positioned 10 pixels away from the left of its original position.
Example

Table 1-2  Move ruler right by 10 pixels

'VBS368
Sub OnClick(ByVal Item)
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
ctrl.MoveRuler 0, 1, 10
End Sub

In the example, the ruler is moved by 10 pixels, starting at reference point 1 (current ruler position). The ruler is now positioned 10 pixels away from the right of its original position.

Example

Table 1-3  Move ruler to end on opening of the window

'VBS369
Sub OnOpen()
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
ctrl.MoveRuler 0, 2, 0
End Sub

In the example, the ruler is moved by 0 pixels, starting at reference point 2 (time axis end position). The ruler is now positioned at the time axis end position.

Example

Table 1-4  Calculate current ruler position

'VBS370
Sub OnClick(ByVal Item)
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim pos
pos = ctrl.MoveRuler (0, 1, 0)
HmiRuntime.Trace "RulerPosition=" & pos & vbCrLf
End Sub

In the example, the ruler is moved by 0 pixels, starting at reference point 1 (current ruler position). The ruler remains in its original position. The ruler position is returned as value.
MoveToFirst method

Function

Executes the "First line" key function of the control.

Syntax

Expression.MoveToFirst()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-

MoveToFirstLine method

Function

Executes the "First message" key function of the AlarmControl.

Syntax

Ausdruck.MoveToFirstLine()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-

MoveToFirstPage method

Function

Executes the "First page" key function of the AlarmControl.

Syntax

Ausdruck.MoveToFirstPage()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToLast method

Function
Executes the "Last data record" key function of the control.

Syntax
Ausdruck.MoveToLast()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToLastLine method

Function
Executes the "Last message" key function of the AlarmControl.

Syntax
Ausdruck.MoveToLastLine()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
MoveToLastPage method

Function
Executes the "Last page" key function of the AlarmControl.

Syntax
Ausdruck.MoveToLastPage()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToNext method

Function
Executes the "Next data record" key function of the control.

Syntax
Ausdruck.MoveToNext()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToNextLine method

Function
Executes the "Next message" key function of the AlarmControl.

Syntax
Ausdruck.MoveToNextLine()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToNextPage method

Function
Executes the "Next page" key function of the AlarmControl.

Syntax
Ausdruck.MoveToNextPage()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

MoveToPrevious method

Function
Executes the "Previous data record" key function of the control.

Syntax
Ausdruck.MoveToPrevious()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
MoveToPreviousLine method

Function

Executes the "Previous message" key function of the AlarmControl.

Syntax

Ausdruck.MoveToPreviousLine()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

MoveToPreviousPage method

Function

Executes the "Previous page" key function of the AlarmControl.

Syntax

Ausdruck.MoveToPreviousPage()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

1.14.5.5   Methods N to R

NextColumn method

Function

Executes the "Next column" key function of the OnlineTableControl.
Syntax

    Ausdruck.NextColumn()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

   -

NextTrend method

Function
Executes the "Next curve" key function of the OnlineTrendControl and FunctionTrendControl.

Syntax

    Ausdruck.NextTrend()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

   -

OneToOneView method

Function
Executes the "Original view" key function of the OnlineTrendControl and FunctionTrendControl.

Syntax

    Ausdruck.OneToOneView()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

   VARIANT
PasteRows method

Function
Executes the "Paste Rows" key function of the UserArchiveControl.

Syntax
Expression.PasteRows()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

PreviousColumn method

Function
Executes the "Previous column" key function of the OnlineTableControl.

Syntax
Ausdruck.PreviousColumn()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

PreviousTrend method

Function
Executes the "Previous curve" key function of the OnlineTrendControl and FunctionTrendControl.

Syntax
Ausdruck.PreviousTrend()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

Print method

Function
Executes the "Print" key function of the control.

Syntax
Ausdruck.Print()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

QuitHorn method

Function
Executes the "Acknowledge central signaling devices" key function of the AlarmControl.

Syntax
Ausdruck.QuitHorn()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
QuitSelected method

Function
Executes the "Single acknowledgment" key function of the AlarmControl.

Syntax
Ausdruck.QuitSelected()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

QuitVisible method

Function
Executes the "Group acknowledgment" key function of the AlarmControl.

Syntax
Ausdruck.QuitVisible()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

Read Method

Description of Tag Object
Reads out the status of a tag (tag object) shortly after the moment it was called. At the same time, the tag object is provided with the values read. Upon reading a tag, its value, quality code and time stamp are determined. The "LastError" property can be used to determine whether the call was successful.

The "Name", "ServerPrefix" and "TagPrefix" properties are not changed as a result.
If the value of the tag is read successfully, the properties of the tag object are assigned the following values:

<table>
<thead>
<tr>
<th>Property</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Tag values</td>
</tr>
<tr>
<td>Name</td>
<td>Tag name (unchanged)</td>
</tr>
<tr>
<td>QualityCode</td>
<td>Quality level</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Current tag time stamp</td>
</tr>
<tr>
<td>LastError</td>
<td>0</td>
</tr>
<tr>
<td>ErrorDescription</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

If the value of the tag is not read successfully, the properties of the tag object are assigned the following values:

<table>
<thead>
<tr>
<th>Property</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>VT_Empty</td>
</tr>
<tr>
<td>Name</td>
<td>Tag name (unchanged)</td>
</tr>
<tr>
<td>QualityCode</td>
<td>Bad Out of Service</td>
</tr>
<tr>
<td>Timestamp</td>
<td>0</td>
</tr>
<tr>
<td>LastError</td>
<td>Read operation error codes</td>
</tr>
<tr>
<td>ErrorDescription</td>
<td>Error description on LastError</td>
</tr>
</tbody>
</table>

**Note**

A summary of possible Quality Codes may be found in WinCC Information System under key word "Communication" > "Diagnostics" or "Communication" > "Quality Codes".

**syntax**

Expression.Read([Readmode])

**Expression**

Necessary. An expression which returns a tag object. The return value of the Read method is the value of the tag read out.

**Parameters**

The optional "Readmode" parameter enables the distinction between two types of reading:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The tag value is read from the process image (cache). 0 is the default value.</td>
</tr>
<tr>
<td>1</td>
<td>The value of a tag is read directly from AS or channel (direct).</td>
</tr>
</tbody>
</table>
If the "Readmode" parameter is omitted, the value is read from the process image by default. The return value of the Read method is the tag value read out as VARIANT.

Reading From the Process Image

When reading from the process image, the tag is logged on and, from that moment, polled cyclically from the PLC. The login cycle is dependent on the configured trigger. The value is read from the tag image by WinCC. For Close Picture, the tag actions are ended again. The call is characterized by the following:

- The value is read by WinCC from the tag image.
- The call is faster in comparison to direct reading (except with the first call: The first call basically takes longer because the value from the PLC must be read out and logged on.)
- The duration of the call is not dependent on the bus load or AS.

Behavior in actions with a tag trigger

All of the tags contained in the tag trigger are already known with Open Picture and are registered with the defined monitoring time. Since all tags are requested at once, the best possible optimization can be targeted from the channel. If a tag, contained in the trigger, is requested with Read during an action, the value already exists and is transferred directly to the call. If a tag is requested which is not contained in the trigger, the behavior is the same as with a standard trigger.

Behavior in actions with a cyclic trigger

tags are registered with half of the cycle time with the first call. For every other call, the value is present.

Behavior in event-driven actions

The tag is registered in the "upon change" mode with the first call. Process tags that are registered in the "upon change" mode correspond with a cyclic read job with a cycle time of 1s.

If an event (e.g. mouse click) requests a value asynchronously, the tag is transferred to the tag image. The tag is requested cyclically from the AS as of this point in time and therefore increases the basic load. To bypass this increase in the basic load, the value can also be read synchronously. The synchronous call causes a one-off increase in the communication load but the tag is not transferred to the tag image.

Direct reading

In the case of direct reading, the current value is returned. The tag is not registered cyclically, the value is requested from the AS one time only. Direct reading has the following properties:

- The value is read explicitly from the AS.
- The call takes longer compared to reading from the process image.
- The duration of the call is dependent on the bus load and AS, amongst other things.
Example:

Reading a tag directly from AS or channel

'VBS100
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read(1)  'Read direct
MsgBox vntValue

Reading a tag from the process image

'VBS101
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read    'Read from cache
MsgBox vntValue

Description of TagSet Object

The TagSet object offers the option of reading several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression

Necessary. An expression which returns an object of type "TagSet".

Reading From the Process Image

The TagSet object offers the advantage of requesting several tags in one read command. The tags are registered in the process image as a group, improving performance in the process.

Direct reading

Since one call may process several read commands, performance is enhanced in comparison to single calls.
Example:

The following example shows how tags are included in the TagSet list, how tag values are imported and subsequently read.

'VB174
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Read
HMIRuntime.Trace "Motor1: " & group("Motor1").Value & vbNewLine
HMIRuntime.Trace "Motor2: " & group("Motor2").Value & vbNewLine

If the optional parameter "Readmode" is set to 1, the process tags are not registered but read directly from AS or channel.

group.Read 1

See also

Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
LastError Property (Page 459)
ErrorDescription Property (Page 411)
TagSet Object (List) (Page 141)
Tag Object (Page 137)

Read Tags method

Function

Executes the "Read tags" key function of the UserArchiveControl.

Syntax

Ausdruck.ReadTags()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

Refresh Method

Function

Drawing all visible pictures again.

Syntax

Expression.Refresh

Expression

Necessary. An expression which returns a "Screens" or "Screen" type object.

Parameters

Examples

The first example forces all visible pictures to be drawn again:

'VBS149
HMIRuntime.Screens.Refresh

The second example forces the basic picture to be immediately redrawn:

'VBS150
HMIRuntime.Screens(1).Refresh

See also

Screen Object (Page 131)
Screens Object (List) (Page 134)
HMIRuntime Object (Page 118)
Remove Method

Description of TagSet Object
Removes a tag from the TagSet list. The tag may be removed by name or reference to a tag object.

syntax
Expression.Remove [Tag]

Expression
Necessary. An expression which returns an object of type "TagSet".

Parameters
VARIANT

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Name of a WinCC tag or reference to a tag object to be removed from the list.</td>
</tr>
</tbody>
</table>

Example:
The following example shows how several tags are included in the TagSet list, and how to remove a tag again.

```vbs
'VBS175
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Remove "Motor1"
```

Description of DataSet Object
Deletes the element specified in parameter "Name" from a list.

syntax
Expression.Remove [Name]

Expression
Necessary. An expression which returns an object of type "DataSet".

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the object to be removed from the list.</td>
</tr>
</tbody>
</table>

Example:

The example shows how to remove the object "motor1" from the list.

'VBS166
HMIRuntime.DataSet.Remove("motor1")

Description of objects Logging, AlarmLogs, DataLogs

The method deletes a previously swapped archive segment from the Runtime project.

Archive segments deleted with the "Remove" method are removed from the common archiving directory of the project.

The call may require a somewhat longer time period, depending on archive data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a triggering tag.

The archive separation and deletion creates a CPU load. This will affect performance.

Note

Calling up the "Remove" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

For redundancy, the following applies: Re-swapped archives are deleted with the "Remove" method only on the computer from which the method was initiated.

Syntax

Objects Logging, AlarmLogs

Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [ServerPrefix]

Expression

Necessary. An expression which returns an object of type "Logging" or "AlarmLogs".

Object DataLogs

Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [Type] [ServerPrefix]
Expression
Necessary. An expression which returns an object of type "DataLogs".

Parameters

**TimeFrom**
Point in time, from which the archives are to be deleted.
When indicating the time format, a short form is also possible. This is described in the "Time Format" section.

**TimeTo**
Time up to which archive segments are to be deleted.
When indicating the time format, a short form is also possible. This is described in the "Time Format" section.

**Timeout**
Timeout in milliseconds.
If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

**Type:**
Type of archive.
The parameter can (optionally) be used only to delete archive segments of the tag logging.
The following values can be entered:

<table>
<thead>
<tr>
<th>Assigned Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hmiDataLogFast</td>
<td>Tag Logging Fast data</td>
</tr>
<tr>
<td>2</td>
<td>hmiDataLogSlow</td>
<td>Tag Logging Slow data</td>
</tr>
<tr>
<td>3</td>
<td>hmiDataLogAll</td>
<td>Tag Logging Fast and Slow data</td>
</tr>
</tbody>
</table>

**ServerPrefix**
Reserved for future versions.

Return value
If an error occurred during deletion of the archive segments, the method will return an error message. Additional information may be found under the subject heading "Error Messages from Database Area".

**Time format**
Time format is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the
information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the statement may be in the form of "YYYY-MM" or "YYYY-MM-DD hh". Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all archive segments between September 2004 up to and including October 4th are to be swapped.

Example:

In the following example, archive segments re-swapped after the fact for a specified time period may be removed and the return value may be output as Trace.

'VBS182

In the following example, all archive segments re-swapped after the fact may be removed and the return value may be output as Trace.

'VBS183
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("","",-1) & vbNewLine

See also

Error Messages from Database Area (Page 820)
Example: How to Start an Action on the Server (Logging Object) (Page 836)
Logging Object (Page 122)
DataSet Object (List) (Page 116)
DataLogs Object (Page 114)
AlarmLogs Object (Page 112)
TagSet Object (List) (Page 141)

RemoveAll Method

Description of TagSet Object

Deletes all tags from a TagSet list.

syntax

Expression.RemoveAll
**Expression**
Necessary. An expression which returns an object of type "TagSet".

**Parameters**

```
--
```

**Example:**
The following example shows how several tags are included in the TagSet list, and how to remove all tags again.

```
'VBS176
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.RemoveAll
```

**Description of DataSet Object**
Deletes all values or object references from a DataSet list.

**Syntax**

```
Expression.RemoveAll
```

**Expression**
Necessary. An expression which returns an object of type "DataSet".

**Parameters**

```
--
```

**Example:**
The example shows how all objects are removed from the list.

```
'VBS167
HMIRuntime.DataSet.RemoveAll
```
See also

- DataSet Object (List) (Page 116)
- TagSet Object (List) (Page 141)
- Tag Object (Page 137)

RemoveData method

Function

Deletes the data of the called trend.

Syntax

```
Expression.RemoveData
```

Expression

Necessary. An expression which returns an object of the "Trend" type.

Example

```
'VBS310
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
objTrend.RemoveData
```

Restore Method

Description of objects Logging, AlarmLogs, DataLogs

The method adds swapped archive segments to the Runtime project.

Upon swapping, the archive segments are copied to the common archiving directory of the project. Therefore, the appropriate storage capacity must be available.

The call may require a somewhat longer time period, depending on archive data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a triggering tag.

Linking / copying of the archives generates a CPU load because the SQL server experiences additional load because of turned-on signature checking in particular. Copying of archive segments will slow down hard disk access.
Upon turned-on signature checking, an error message is returned if an unsigned or modified archive is to be swapped. There is always only one error message returned, even if several errors occurred during the swap process. Additionally, a WinCC system message is generated for each archive segment. An entry is added to the Windows event log in the "Application" section. This provides the opportunity to check which archive segments are creating the error.

- With an unsigned archive, the return value "0x8004720F" is returned. The archive is stored. The following text is entered in the event display: "Validation of database <db_name> failed! No signature found!"
- With an changed archive, the return value "0x80047207" is returned. The even screen, the entry is "Validation of database <db_name> failed!". The archive is not stored.

**Note**
Calling up the "Restore" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

For redundancy, the following applies: Upon re-swapping of archives with the "Restore" method, only archive segments are added to the Runtime project on the computer from which the method was called.

**Syntax**

**Objects Logging, AlarmLogs**

Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut] [ServerPrefix]

*Expression*

Required. An expression which returns an object of type "Logging" or "AlarmLogs".

**Object DataLogs**

Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut] [Type] [ServerPrefix]

*Expression*

Required. An expression which returns an object of type "DataLogs".

**Parameter**

*SourcePath*

Path to archive data.
**TimeFrom**
Point in time, from which the archives are to be stored.
When indicating the time format, a short form is also possible. This is described in the "Time Format" section.

**TimeTo**
Time up to which archive segments are to be swapped.
When indicating the time format, a short form is also possible. This is described in the "Time Format" section.

**Timeout**
Timeout in milliseconds.
If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

**Type**
Type of archive.
The parameter can (optionally) be used only to store archive segments of the tag logging. The following values can be entered:

<table>
<thead>
<tr>
<th>Assigned Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hmiDataLogFast</td>
<td>Tag Logging Fast data</td>
</tr>
<tr>
<td>2</td>
<td>hmiDataLogSlow</td>
<td>Tag Logging Slow data</td>
</tr>
<tr>
<td>3</td>
<td>hmiDataLogAll</td>
<td>Tag Logging Fast and Slow data</td>
</tr>
</tbody>
</table>

**ServerPrefix**
Reserved for future versions.

**Return value**
If an error occurred during swapping of archive segments, the method will return an error message. Additional information may be found under the subject heading "Error Messages from Database Area".

**Time format**
Time format is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the statement may be in the form of "YYYY-MM" or "YYYY-MM-DD hh". Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all archive segments between September 2004 up to and including October 4th are to be swapped.
Example

In the following example, all archive segments since the specified time period are re-swapped, and the return value is output as Trace.

'VBS184
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Folder","2004-09-14","",-1) & vbCrLf

In the following example, all Tag Logging Slow archive segments since the specified time period are re-swapped, and the return value is output as Trace.

'VBS185
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:\Folder","2004-09-14 12:30:05","2004-09-20 18:30",-1,2) & vbCrLf

In the following example, all Alarm Logging archive segments up to the specified time period are re-swapped, and the return value is output as Trace.

'VBS186

See also

Error Messages from Database Area (Page 820)
Example: How to Start an Action on the Server (Logging Object) (Page 836)
Logging Object (Page 122)
DataLogs Object (Page 114)
AlarmLogs Object (Page 112)

1.14.5.6 Methods S to T

SelectAll

Function

Selects all rows in the table-based control.
Syntax

```
Expression.SelectAll()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

See also

Row object (list) (Page 225)

**SelectRow**

Function

Selects a particular row in the table-based control.

Syntax

```
Expression.SelectRow(ByVal IRow As Long, Optional bExtendSelection As Boolean)
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRow</td>
<td>Number of the row to be selected.</td>
</tr>
<tr>
<td>bExtendSelection</td>
<td>Indicates as an option whether the current selection will be extended. Is only relevant if multiple selections are possible.</td>
</tr>
</tbody>
</table>

Example

- Row 1 is currently selected. If SelectRow(2, True) is called, then row 1 and row 2 will be selected.
- Row 1 is currently selected. If SelectRow(2, False) or SelectRow(2) is called without an optional parameter, then only row 2 will be selected.

See also

Row object (list) (Page 225)
**SelectedStatisticArea method**

**Function**

Executes the "Set statistic area" key function of the OnlineTableControl.

**Syntax**

```
Ausdruck.SelectedStatisticArea()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**

- -

**ServerExport method**

**Function**

Executes the "Export archive" key function of the UserArchiveControl.

**Syntax**

```
Ausdruck.ServerExport()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**

- -

**ServerImport method**

**Function**

Executes the "Import archive" key function of the UserArchiveControl.

**Syntax**

```
Ausdruck.ServerImport()
```
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ShowColumnSelection method

Function
Executes the "Select columns" key function of the OnlineTableControl.

Syntax
Ausdruck.ShowColumnSelection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ShowComment method

Function
Executes the "Comments dialog" key function of the AlarmControl.

Syntax
Ausdruck.ShowComment()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
ShowDisplayOptionsDialog method

Function

Executes the "Display options dialog" key function of the AlarmControl.

Syntax

Ausdruck.ShowDisplayOptionsDialog()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowEmergencyQuitDialog method

Function

Executes the "Emergency acknowledgment" key function of the AlarmControl.

Syntax

Ausdruck.ShowEmergencyQuitDialog()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowHelp method

Function

Executes the "Help" key function of the control.

Syntax

Ausdruck.ShowHelp()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
VARIANT

ShowHideList method

Function
Executes the "List of messages to be hidden" key function of the AlarmControl.

Syntax
Ausdruck.ShowHideList()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ShowHitList method

Function
Executes the "Hitlist" key function of the AlarmControl.

Syntax
Ausdruck.ShowHitList()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
ShowInfoText method

Function

Executes the "Info text dialog" key function of the AlarmControl.

Syntax

Ausdruck.ShowInfoText()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

-

ShowInsertValueDialog method

Function

Executes the "Create archive value" key function of the OnlineTableControl.

Syntax

Expression.ShowInsertValueDialog()

Expression

Required. An expression that returns an object of the "ScreenItem" type.

ShowLockDialog method

Function

Executes the "Lock dialog" key function of the AlarmControl.

Syntax

Ausdruck.ShowLockDialog()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.
Parameters

ShowLockList method

Function
Executes the "Lock list" key function of the AlarmControl.

Syntax
Ausdruck.ShowLockList()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

ShowLongTermArchiveList method

Function
Executes the "Long-term archive list" key function of the AlarmControl.

Syntax
Ausdruck.ShowLongTermArchiveList()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

ShowMessageList method

Function
Executes the "Message list" key function of the AlarmControl.
Syntax

Ausdruck.ShowMessageList()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ShowPercentageAxis method

Function
Executes the "Relative axis" key function of the OnlineTrendControl.

Syntax

Ausdruck.ShowPercentageAxis()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ShowPropertyDialog method

Function
Executes the "Configuration dialog" key function of the control.

Syntax

Ausdruck.ShowPropertyDialog()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
VARIANT
ShowSelectArchive method

Function

Executes the "Select data connection" key function of the UserArchiveControl.

Syntax

Ausdruck.ShowSelectArchive()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowSelection method

Function

Executes the "Selection dialog" key function of the UserArchiveControl.

Syntax

Ausdruck.ShowSelection ()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowSelectTimeBase method

Function

Executes the "Time base dialog" key function of the UserArchiveControl.

Syntax

Ausdruck.ShowSelectTimeBase()
Expression
Necessary. An expression that returns an object of the “ScreenItem” type.

Parameters
- -

ShowSelectionDialog method

Function
Executes the "Selection dialog" key function of the AlarmControl.

Syntax
Ausdruck.ShowSelectionDialog()

Expression
Necessary. An expression that returns an object of the “ScreenItem” type.

Parameters
- -

ShowShortTermArchiveList method

Function
Executes the "Short-term archive list" key function of the AlarmControl.

Syntax
Ausdruck.ShowShortTermArchiveList()

Expression
Necessary. An expression that returns an object of the “ScreenItem” type.

Parameters
- -
ShowSort method

Function

Executes the "Sort dialog" key function of the UserArchiveControl.

Syntax

Ausdruck.ShowSort()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowSortDialog method

Function

Executes the "Sort dialog" key function of the AlarmControl.

Syntax

Ausdruck.ShowSortDialog()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

ShowTagSelection method

Function

Executes the "Select data connection" key function of the control.

Syntax

Ausdruck.ShowTagSelection()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
--

ShowTimebaseDialog method

Function
Executes the "Time base dialog" key function of the AlarmControl.

Syntax
Ausdruck.ShowTimebaseDialog()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
--

ShowTimeSelection method

Function
Executes the "Select time range" key function of the control.

Syntax
Ausdruck.ShowTimeSelection()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
--
ShowTrendSelection method

Function

Executes the "Select trends" key function of the OnlineTrendControl and FunctionTrendControl.

Syntax

Ausdruck.ShowTrendSelection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

---

StartStopUpdate method

Function

Executes the "Start" or "Stop" key function of the control.

Syntax

Ausdruck.StartStopUpdate()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

---

Stop Method

Function

Terminates WinCC Runtime.

Syntax

HMIRuntime.Stop
Parameters

Example:

The following example terminates WinCC Runtime:

'SBS124
   HMIRuntime.Stop

See also

HMIRuntime Object (Page 118)

Trace Method

Description

Displays messages in the diagnostics window.

syntax

HMIRuntime.Trace

Parameters

STRING

Example:

The following example writes a text in the diagnostics window:

'SBS103
   HMIRuntime.Trace "Customized error message"

See also

HMIRuntime Object (Page 118)
1.14.5.7 Methods U to Z

UnhideAlarm method

Function

Executes the "Unhide alarm" key function of the AlarmControl.

Syntax

Ausdruck.UnhideAlarm()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

UnlockAlarm method

Function

Executes the "Unlock alarm" key function of the AlarmControl.

Syntax

Ausdruck.UnlockAlarm()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

UnselectAll

Function

Deselects all rows in the table-based control.
Syntax

```
Expression.UnselectAll()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

### Parameters

- -

### See also

Row object (list) (Page 225)

---

### UnselectRow

**Function**

Deselects a particular row in the table-based control.

**Syntax**

```
Expression.UnselectRow(ByVal IRow As Long)
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

### Parameters

Long

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRow</td>
<td>Number of the row to be selected.</td>
</tr>
</tbody>
</table>

### See also

Row object (list) (Page 225)

---

### Write Method

**Description of Tag Object**

Writes a value synchronously or asynchronously in a tag. The "LastError" property can be used to determine whether the call was successful.
If the value of the tag is set successfully, the properties of the tag object are assigned the following values:

<table>
<thead>
<tr>
<th>Property</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Tag values set by the user (unchanged)</td>
</tr>
<tr>
<td>Name</td>
<td>Tag name (unchanged)</td>
</tr>
<tr>
<td>QualityCode</td>
<td>Bad Out of Service</td>
</tr>
<tr>
<td>Timestamp</td>
<td>0</td>
</tr>
<tr>
<td>LastError</td>
<td>0</td>
</tr>
<tr>
<td>ErrorDescription</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

If the value of the tag is not set successfully, the properties of the tag object are assigned the following values:

<table>
<thead>
<tr>
<th>Property</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Tag values set by the user (unchanged)</td>
</tr>
<tr>
<td>Name</td>
<td>Tag name (unchanged)</td>
</tr>
<tr>
<td>QualityCode</td>
<td>Bad Out of Service</td>
</tr>
<tr>
<td>Timestamp</td>
<td>0</td>
</tr>
<tr>
<td>LastError</td>
<td>Write operation error codes</td>
</tr>
<tr>
<td>ErrorDescription</td>
<td>Error description on LastError</td>
</tr>
</tbody>
</table>

**syntax**

```
Expression.Write [Value],[Writemode]
```

**Expression**

Necessary. An expression which returns a tag object.

**Parameters**

The value to be written can be transferred directly to the method as a parameter. If the parameter is not specified, the value in the "Value" property is used. The "Writemode" option parameter can be used to select whether the tag value should be written synchronously or asynchronously. If the "Writemode" parameter is not used, writing is performed asynchronously as its default value.

During the writing process, no information is supplied on the status of the tags.
The "Value" property contains the value which was set before or during the writing operation, therefore is may not correspond to the real current value of the tag. If the data on the tag should be updated, use the Read method.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value (optional)</td>
<td>The tag value is specified. The specified value overwrites the value in the &quot;Value&quot; property in the tag object. The tag value is not specified. The tag receives the current value from the &quot;Value&quot; property of the tag object.</td>
</tr>
<tr>
<td>Writemode (optional)</td>
<td>0 or empty: The tag value is written asynchronously. 0 is the default value. 1: The tag value is written synchronously.</td>
</tr>
</tbody>
</table>

On asynchronous writing, it is written immediately into the tag image. The user does not receive any feedback if the value has been written in the programmable controller, too.

In the case of synchronous writing (direct to the PLC), the writing operation actually occurs when the PLC is ready to operate. The use receives a check-back message if the writing operation was not successful.

**Example:**

**Asynchronous writing**

'VBS104
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write
MsgBox objTag.Value

or

'VBS105
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5
MsgBox objTag.Value
Synchronous writing

'VBS106
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write ,1
MsgBox objTag.Value

or

'VBS107
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5, 1
MsgBox objTag.Value

Description of TagSet Object

The TagSet object offers the option of writing several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression
Necessary. An expression which returns an object of type "TagSet".

Parameters
In order to write different values, the "Value" property of individual tag objects must be set, and write must be called thereafter without the "Value" parameter. Since the write commands are grouped into one call, it results in improved performance compared to single calls.

In a TagSet object, it is not possible to pass on a value using the "Write" method. Individual values must be set using the "Value" property of the individual tag objects.
Example:

The following example shows how tags are included in the TagSet list, how tag values are set and subsequently written.

'VBS173
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Wert1"
group.Add "Wert2"
group("Wert1").Value = 3
group("Wert2").Value = 9
group.Write

group.Write 1

If you set the optional parameter "Writemode" equal to 1, the process tags are written synchronously (directly to AS).

See also

LastError Property (Page 459)
ErrorDescription Property (Page 411)
TagSet Object (List) (Page 141)
Tag Object (Page 137)

WriteTags method

Function

Executes the "Write tags" key function of the UserArchiveControl.

Syntax

Expression.WriteTags()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -
**ZoomArea - Method**

**Function**
Executes the “Zoom area” key function of the OnlineTrendControl and FunctionTrendControl.

**Syntax**
Ausdruck.ZoomArea()

**Expression**
Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**
--

**ZoomInOut - Method**

**Function**
Executes the “Zoom +/-” key function of the OnlineTrendControl and FunctionTrendControl.

**Syntax**
Ausdruck.ZoomInOut()

**Expression**
Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**
--

**ZoomInOutTime method**

**Function**
Executes the "Zoom time axis +/-" key function of the OnlineTrendControl.

**Syntax**
 Ausdruck.ZoomInOutTime()
Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ZoomInOutValues - Method

Function
Executes the "Zoom value axis +/-" key function of the OnlineTrendControl.

Syntax
Ausdruck.ZoomInOutValues()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -

ZoomInOutX method

Function
Executes the "Zoom X axis +/-" key function of the FunctionTrendControl.

Syntax
Ausdruck.ZoomInOutX()

Expression
Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters
- -
### ZoomInOutY - Method

**Function**

Executes the "Zoom Y axis +/-" key function of the FunctionTrendControl.

**Syntax**

```vbs
Ausdruck.ZoomInOutY()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**

- -

### ZoomMove method

**Function**

Executes the "Move trend area" key function of the OnlineTrendControl and FunctionTrendControl.

**Syntax**

```vbs
Ausdruck.ZoomMove()
```

**Expression**

Necessary. An expression that returns an object of the "ScreenItem" type.

**Parameters**

- -

### Appendix

#### 1.14.6 Error Messages from Database Area

**Introduction**

Upon access to databases, a value is returned upon execution. Values in the range "0x8..." represent an error message. Values not equal to "0x8..." represent a status message.
Status Messages

The following status messages are defined:

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>OK</td>
</tr>
</tbody>
</table>
| 0x1  | Function did not find any errors in parameter supply and did not find any internal errors. The following causes may result in this value. When connecting databases:  
- No archive could be found in the given time window.  
- Archives were found in the given time window, but they were already connected. When separating databases:  
- No connected archives could be found in the given time window. No checks are performed on whether or not archives are attached at all. |

Error Messages

The following error messages are defined (n in English only):

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80047200</td>
<td>WinCC is not activated</td>
</tr>
<tr>
<td>0x80047201</td>
<td>Invalid archive type</td>
</tr>
<tr>
<td>0x80047202</td>
<td>Invalid lower boundary</td>
</tr>
<tr>
<td>0x80047203</td>
<td>Invalid upper boundary</td>
</tr>
<tr>
<td>0x80047204</td>
<td>Path 'CommonArchiving' could not be created in the project path</td>
</tr>
<tr>
<td>0x80047205</td>
<td>Timeout, please retry</td>
</tr>
<tr>
<td>0x80047206</td>
<td>WinCC was deactivated</td>
</tr>
</tbody>
</table>
| 0x80047207 | Wrong signification  
At least one database had an invalid signature and has not been attached. |
| 0x80047208 | Database could not be attached |
| 0x80047209 | Copy to 'CommonArchiving' is not possible. |
| 0x8004720A | Invalid syntax for database filename. |
| 0x8004720B | No list of databases. |
| 0x8004720C | Database already detached. |
| 0x8004720D | Database could not be detached. |
| 0x8004720F | Unsigned database attached.  
At least one database without signature has been attached. |
| 0x80047210 | Path error:  
- Path invalid,  
- no *.MDF files found in specified path or  
- no permission to specified path. |

See also

Remove Method (Page 789)
Write Method (Page 813)
Read Method (Page 783)
Restore Method (Page 794)
Logging Object (Page 122)
DataLogs Object (Page 114)
AlarmLogs Object (Page 112)
1.15 Examples of VBScript

1.15.1 Examples of VBScript

Introduction

The following section contains application examples of VBS in WinCC. The "Examples in WinCC" section contains examples of codes with which the WinCC Runtime environment can be made dynamic. These examples have been conceived so that they can be assumed 1:1 in the configuration.

The "General Examples" section contains examples with which to influence the Microsoft environment. There is no guarantee nor support for the running capability of these examples.

See also

Examples in WinCC (Page 823)

1.15.2 Examples in WinCC

1.15.2.1 Examples in WinCC

Introduction

This section contains examples of using VBScript in WinCC with regard to the following topics:

- Access to objects in the Graphics Designer (e.g. color or text change)
- Set color of objects above RGB colors
- Configuring language change
- Deactivate Runtime
- Start external program
- Globally configure picture change (from Global Script)
- Configuring Change Picture Via Property
- Use trace for diagnostics output
- Set value of a tag
- Read value of a tag
- Check the success of a read/write action into a tag
- Asynchronously set value of a tag
1.15 Examples of VBScript

See also

Example: Starting an external application (Page 859)
Example: Writing Object Properties (Page 833)
Example: How to Read Tag Values (Page 830)
Example: Writing tag values (Page 828)
Example: Configuring diagnostics output via Trace (Page 827)
Example: Configuring Change Picture Via Property (Page 827)
Example: Configuring change picture globally (Page 826)
Example: Deactivating Runtime (Page 826)
Example: How to Configure Language Changes (Page 825)
Example: Defining the color of objects (Page 825)
Example: Accessing objects in Graphics Designer (Page 824)

1.15.2.2 Example: Accessing objects in Graphics Designer

Introduction

Access can be made to all Graphic Designer objects using VBS WinCC in order to make the graphic Runtime environment dynamic. Graphic objects can be made dynamic on operation (e.g. clicking the mouse on a button), depending on a tag or cyclically (e.g. flashing).

The following examples illustrate how to change a graphic object following a mouse click.

Procedure

In the following example, the radius of a circle is set to 20 in Runtime per mouse click:

'VBS121
Dim objCircle
Set objCircle= ScreenItems("Circle1")
objCircle.Radius = 20

Note

The expression used in the example only applies to Graphics Designer. In the case of analog actions in Global Script, address the objects using the HMIRuntime object.

See also

Examples in WinCC (Page 823)
1.15.2.3 Example: Defining the color of objects

Introduction

The colors of graphic objects are defined via RGB values (Red/Green/Blue). The color values for graphic objects can be set or read out.

Procedure

The following example defines the fill color for "ScreenWindow1" to blue:

'VBS122
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.FillStyle = 131075
objScreen.FillColor = RGB(0, 0, 255)

See also

Examples in WinCC (Page 823)

1.15.2.4 Example: How to Configure Language Changes

Introduction

The Runtime language of WinCC can be changed using VBS. The most typical use is buttons with the corresponding language codes which are placed on the start page of a project.

You specify the Runtime language in VBS by using a country code, e.g., 1031 for German - Default, 1033 for English - USA etc. A summary of all country codes may be found in the Basics of VBScript under the subject header "Regional Scheme ID (LCID) Diagram".

Procedure

Use the "Mouse click" event on a button to create a VBS action and enter the following action code to switch the Runtime language to German:

'VBS123
HMIRuntime.Language = 1031

See also

Examples in WinCC (Page 823)
1.15.2.5 Example: Deactivating Runtime

Introduction

It is possible to terminate WinCC Runtime with VBS, e.g. via a mouse click or in dependence on tag values or other events, such as multiple faulty input of a password when starting Runtime.

What to do

The following example terminates WinCC Runtime:

'VBS124
HMIRuntime.Stop

See also

Examples in WinCC (Page 823)

1.15.2.6 Example: Configuring change picture globally

Introduction

VBS can be used to initiate a global picture change and thus, for example, display a picture from a server on a client in a distributed system. To do this, server's server prefix must precede the target picture.

What to do

Configure the following code for a picture change to a button, for example:

'VBS125
HMIRuntime.BaseScreenName = "Serverprefix::New screen"

See also

Examples in WinCC (Page 823)
1.15.2.7 Example: Configuring Change Picture Via Property

Introduction

If partitioned pictures are used in the configuration, e.g. in a basic picture title and operating bar for the user interface and an embedded picture window for the actual picture display, configure a picture change using the properties of the picture window.

The property of the "ScreenName" picture window must be changed in order for the other picture to appear. The action and picture window must be configured in the same picture.

What to do

In the following example, the "test.pdl" picture is displayed in the "ScreenWindow" picture window when executing the action:

''VBS126
Dim objScrWindow
Set objScrWindow = ScreenItems("ScreenWindow")
objScrWindow.ScreenName = "test"

See also

Examples in WinCC (Page 823)

1.15.2.8 Example: Configuring diagnostics output via Trace

Introduction

If a GSC diagnostics window has been inserted in the picture, diagnostics output can be displayed in the diagnostics window in Runtime using the Trace command.

GSC Diagnostics issues the Trace methods contained in the actions in the chronological sequence they are called. This also applies to Trace instructions in procedures which are called in actions. The targeted implementation of Trace instructions, e.g. for the output of tag values, enables the progress of actions and the procedures called in them to be traced. The Trace instructions are entered in the form "HMIRuntime.Trace(<Ausgabe>)".

The GSC Diagnostics displays trace output from C and VBS.

What to do

The following example writes a text in the diagnostics window:

''VBS127
HMIRuntime.Trace "Customized error message"
1.15.2.9 Example: Writing tag values

Introduction

Using VBS, it is possible to write a tag value to the PLC, e.g. by clicking the mouse on a button to specify setpoint values, or to set internal tag values to trigger other actions. Various write variations are mentioned and explained below.

Simple writing

In the following example, a value is written to the "Tag1" tag:

'VBS128
HMIRuntime.Tags("Tag1").Write 6

This is the simplest form of writing since no object reference is generated.

Writing with object reference

In the following example, a local copy of the tag object is created and a value written to "Tag1":

'VBS129
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 7

Referencing offers the advantage of being able to work with the tag object before writing. The tag value can be read, calculations executed and written again:

'VBS130
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
Synchronous writing

Normally, the value to be written is transferred to the tag management and processing of the action resumed. In some cases, however, it must be ensured that the value has actually been written before processing of the action can be resumed.

This type of writing is realized by specifying the value 1 for the additional, optional parameters:

\[
\begin{align*}
'VBS131 \\
\text{Dim } \text{objTag} \\
\text{Set } \text{objTag} = \text{HMIRuntime.Tags("Tag1")} \\
\text{objTag.Write 8.1}
\end{align*}
\]

or

\[
\begin{align*}
'VBS132 \\
\text{Dim } \text{objTag} \\
\text{Set } \text{objTag} = \text{HMIRuntime.Tags("Tag1")} \\
\text{objTag.Value = 8} \\
\text{objTag.Write ,1}
\end{align*}
\]

Note

Please note that the call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, amongst other things.

The type of writing complies to the SetTagXXXWait() call in C scripting.

Writing with status handling

In order to ensure that a value has been written successfully, it is necessary to execute an error check or determine the status of the tag, after the writing process.

This is done by checking the value of the "LastError" property after writing. When the test proves successful, i.e. the job has been placed successfully, the tag status is checked.

In the case of a write job, the current status from the process is not determined. To establish this, it is necessary to read the tag. The value specified in the Quality Code property after the read process provides an indication of the tag status and, if necessary, makes reference to a failed AS connection.
In the following example, the "Tag1" tag is written. If an error occurs during writing, the error value and error description appear in the Global Script diagnostics window. Finally, the Quality Code is checked. If the Quality Code is no OK (0x80), it is displayed in the diagnostics window.

'VBS133
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 9
If 0 <> objTag.LastError Then
    HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " & objTag.ErrorDescription & vbCrLf
Else
    objTag.Read
    If &H80 <> objTag.QualityCode Then
        HMIRuntime.Trace "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf
    End If
End If
End If

Note

After writing a tag, the QualityCode property of the local tag object is set to "BAD Out of Service" because it is not known which Quality Code manages the tag in the process.

The Quality Code cannot be written from VBS.

See also

Write Method (Page 813)
Examples in WinCC (Page 823)

1.15.2.10 Example: How to Read Tag Values

Introduction

VBS can be used to read and further process a tag value. This makes it possible, for example, to click the mouse on a button to obtain information on the system status or to execute a calculation.

Various read variations are mentioned and explained below.
**Simple reading**

In the following example, the value of “Tag1” is read and displayed in the Global Script diagnostics window:

```
'VBS134
HMIRuntime.Trace "Value: " & HMIRuntime.Tags("Tag1").Read & vbCrLf
```

This is the simplest form of reading since no object reference is generated.

**Reading with object reference**

In the following example, a local copy of the tag object is created, the tag value read and displayed in the Global Script diagnostics window:

```
'VBS135
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
HMIRuntime.Trace "Value: " & objTag.Read & vbCrLf
```

Referencing offers the advantage of being able to work with the tag object. The tag value can be read, calculations executed and written again:

```
'VBS136
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
```

Using the Read method, process tags which have been read are added to the image, from this moment on they cyclically requested from the AS. If the tag is already in the image, the value contained in it is returned.

For Close Picture, the tag actions are ended again.

**Note**

If a tag is requested in a Global Script action, it remains registered throughout the enter Runtime of WinCC.

**Direct reading**

Normally, the tag values are read from the tag image. In certain situations, however, it may be necessary to read the value direct from the AS, e.g. to synchronize fast processes.
If the optional parameter is set to 1 for the read process, the tag is not logged in cyclically but the value is requested once from the AS.

'VBS137
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
HMIRuntime.Trace "Value: " & objTag.Read(1) & vbCrLf

Note
Please note that the call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, amongst other things.
This type of call must be avoided in the case of cyclic C actions because this is the main reason for performance problems.
This type of read process corresponds to GetTagXXXWait() call from C scripting.

Reading with status handling

In order to ensure that a value is valid, a check should be made following reading. This occurs by the fact that the Quality Code is controlled.

In the following example, the "myWord" tag is read and the QualityCode then checked. When the Quality Code does not correspond to OK (0x80) the LastError, ErrorDescription and QualityCode properties are displayed in the Global Script diagnostics window.

'VBS138
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
If &H80 <> objTag.QualityCode Then
HMIRuntime.Trace "Error: " & objTag.LastError & vbCrLf & "ErrorDescription: " & objTag.ErrorDescription & vbCrLf & "QualityCode: 0x" & Hex(objTag.QualityCode) & vbCrLf
Else
HMIRuntime.Trace "Value: " & objTag.Value & vbCrLf
End If

Note
If an error occurs during reading, QualityCode is set to BAD NON-SPECIFIC. Therefore, it is sufficient to check the QualityCode following reading.
1.15.2.11 Example: Writing Object Properties

Introduction

VBS enables access to the properties of all Graphics Designer picture objects. Properties can be read out to be modified or changed during Runtime.

The following examples illustrate various forms of access.

Simple setting of a property

In the following example, the background color of the "Rectangle1" object contained in the picture is set to red:

'VBS139
ScreenItems("Rectangle1").BackColor = RGB(255,0,0)
This is the simplest form of writing since no object reference is generated.

Note

If the work is completed without an object reference, only the standard properties are provided in Intellisense.

The form of expression used in the example only applies to Graphics Designer. In the case of analog actions in Global Script, address the objects using the HMIRuntime object.

Setting a property with object reference

In the following example, a reference is created to the "Rectangle1" object contained in the picture and the background is set to red using the VBS standard function RGB():

'VBS140
Dim objRectangle
Set objRectangle = ScreenItems("Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
Referencing is useful when several object properties must be changed. When using Intellisense, this process then lists all the object properties.

---

**Note**

The form of expression used in the example only applies to Graphics Designer. In the case of analog actions in Global Script, address the objects using the HMIRuntime object.

---

**Setting properties via the picture window**

VBS in Graphics Designer offers two possibilities for picture transcending addressing:

- via the Screen object of a picture window with "ScreenItems"
- from the basic picture with "HMIRuntime.Screens"

**Referencing via the picture window**

In the following example, the color of a rectangle is changed in an subordinate picture window. The script is executed in the picture "BaseScreen", in which the picture window "ScreenWindow1" is located. The picture window displays a picture, which contains an object of the type "Rectangle" with the name "Rectangle1".

```
'VBS199
Sub OnLButtonUp(ByVal Item, ByVal Flags, ByVal x, ByVal y)
  Dim objRectangle
  Set objRectangle = ScreenItems("ScreenWindow1").Screen.ScreenItems("Rectangle1")
  objRectangle.BackColor = RGB(255,0,0)
End Sub
```

**Referencing from the basic picture**

You can reference the picture with the object to be modified via HMIRuntime.Screens. The specification of the picture is defined relative to the basic picture via the following access code:

```
[<Grundbildname>.]<Bildfenstername>[.<Bildname>]...<Bildfenstername>[.<Bildname>]
```

In the following example, a reference is created to the "Screen2" object contained in the "Rectangle1" picture and the background color is set to red.

The picture "Screen2", in this case, is in "Screen1". "Screen1" is displayed in the basic picture "BaseScreen".

```
'VBS141
Dim objRectangle
Set objRectangle = HMIRuntime.Screens("BaseScreen.ScreenWindow1:Screen1.ScreenWindow1:Screen2").ScreenItems("Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
```
It is not necessary to specify the picture name. It is possible to address a picture uniquely using the picture window name. Therefore, it is sufficient to specify the name of the picture window, as in the following example:

```vbs
'VBS142
Dim objRectangle
Set objRectangle = HMIRuntime.Screens("ScreenWindow1.ScreenWindow2").ScreenItems("Rectangle1")
objRectangle.BackColor = RGB(255,0,0)
```

This type of addressing enables objects in picture windows to be addressed in different pictures. This is a particularly interesting aspect in respect of the picture module technique.

**Make the property dynamic using the return value**

Actions on properties can not only be triggered by events or cyclically but properties can also be made dynamic directly via an action.

In the following example, the background color of an object is made dynamic via a return value. The value transferred can come from the evaluation of events in the PLC, for example and used for the graphic display of an operating status:

```vbs
'VBS146
Function BackColor_Trigger(ByVal Item)
    BackColor_Trigger = RGB(125,0,0)
End Function
```

**Note**

If you make an object property dynamic with a VBS action via the return value of a script, the value of the object property is written only if it has changed in relation to the last script run. It is not considered if the value had been changed from another location. Therefore it is illegal to change properties which have been made dynamic by VBS action via the return value from another location (e.g., other C scripts or VBS scripts).

if you do not observe this, wrong values can be the results.

**See also**

VBS Reference (Page 105)

Examples in WinCC (Page 823)
Introduction

In multi-user projects, the Logging object presently functions on the server only. The following example shows how to start an action on the server from the client, and how to swap and delete archive segments on client accordingly.

The example shows a global action started with a control tag. The contents of the control tag determine whether the "Restore" method or the "Remove" method is called. At the end of the action, the control tag is set to "0".

A query prevents the action from being started on client computers.

Path and time period are passed on by internal tags.

The path information may also contain a network release. Archive segments to be swapped must therefore not be stored locally at the server. It must be warranted, though, that the server may directly access the path.

Note

The example shows a delete suggestion and may be adjusted as needed.

What to do

1. Create the following internal tags with project-wide updating in the WinCC Explorer:
   - StartLogging (unsigned 8 bit value)
   - SourcePath (Text tag 8 bit character set)
   - TimeFrom (Text tag 8 bit character set)
   - TimeTo (Text tag 8 bit character set)
   - RetVal (signed 32 bit value)

2. Create a global VBS action and enter the tag 'StartLogging' as tag trigger with cycle "Upon Change".

3. Copy the following script into the action
'VBS180
Dim StartLogging
Dim SourcePath
Dim TimeFrom
Dim TimeTo
Dim RetVal
'Exit when running on client
If (Left(HMIRuntime.ActiveProject.Path, 1) = "\") Then
  Exit Function
End If
'read parameters
StartLogging = HMIRuntime.Tags("StartLogging").Read
SourcePath = HMIRuntime.Tags("SourcePath").Read(1)
TimeFrom = HMIRuntime.Tags("TimeFrom").Read(1)
TimeTo = HMIRuntime.Tags("TimeTo").Read(1)
'restore or remove depends on the parameter
If (StartLogging = 1) Then
  RetVal = HMIRuntime.Logging.Restore(SourcePath, TimeFrom, TimeTo, -1)
  HMIRuntime.Tags("RetVal").Write RetVal, 1
ElseIf (StartLogging = 2) Then
  RetVal = HMIRuntime.Logging.Remove(TimeFrom, TimeTo, -1)
  HMIRuntime.Tags("RetVal").Write RetVal, 1
End If

The action may be started on a client with the following action, for example. Please note that parameters must be written prior to setting the control tag.

'VBS181
'set parameters
HMIRuntime.Tags("SourcePath").Write "\client_pc\temp",1
HMIRuntime.Tags("TimeFrom").Write "2004",1
HMIRuntime.Tags("TimeTo").Write "2004",1
'start action
HMIRuntime.Tags("StartLogging").Write 1.1

Note
Tags are predominantly written and read in "direct" mode. This will synchronize the sequences. Since this deals with internal tags, this mode may be used without any further concerns.
1.15.2.13 Example: You connect or disconnect a database backup with VBS

Introduction

You can connect and read swapped-out databases with VBScript.

The example below describes access to a swapped-out WinCC process value archive via the WinCC OLE DB Provider. A backup file is connected, read and the connection terminated again.

1. Create connection for access via ADO and specify data source (1.1)
2. Define command for access: Read relative time range for the last 10 minutes (1.2)
3. Establish the connection (2.1)
4. Execute command (2.2)
5. Define data record structure and test first data record (2.3)
6. Call subsequent rows of the result set (3.1)
7. Clean up: Disconnect (4.1)

The example does not include error processing.

Further information on the OLE DB Provider is available in the documentation for the WinCC/Connectivity Pack under "Access via OLE DB Provider > Retrieve archive data".

Requirement

- WinCC or WinCC/Connectivity Pack is installed on the PC.
Example

'VBS372
Sub OnClick(Byval Item)

Const MaxRows = 100

' 1.1 Create connection string for ADO access
Dim ProviderName
Dim CatalogName
Dim DataSourceName
Dim ConnectionString
Dim CommandString

ProviderName = "Provider=WinCCOLEDBProvider.1;"
CatalogName = "Catalog=CC_Test_15_04_14_08_23_27R;"
DataSourceName = "Data Source=\WinCC"
ConnectionString = ProviderName + CatalogName + DataSourceName

' 1.2 Define command string for access (relative time range, last 10 minutes)
CommandString = "TAG:R,'ArcTag\Tag_1','0000-00-00 00:10:00.000','0000-00-00 00:00:00.000'"

Dim DBConnection
Dim DBCommand
Dim DBRecordset

' 2.1 Open connection
HMIRuntime.Trace "before open connection" & vbNewLine
Set DBConnection = CreateObject("ADODB.Connection")
DBConnection.ConnectionString = ConnectionString
DBConnection.CursorLocation = 3
DBConnection.open

Set DBRecordset = CreateObject("ADODB.Recordset")

' 2.2 Execute command
HMIRuntime.Trace "before execute command" & vbNewLine
Set DBCommand = CreateObject("ADODB.Command")
DBCommand.CommandType = 1
DBCommand.ActiveConnection = DBConnection
DBCommand.CommandText = CommandString
Set DBRecordset = DBCommand.Execute

Dim lngCount
Dim lngValue

' 2.3 Record structure and first record data (test)
lngCount = DBRecordset.Fields.Count
HMIRuntime.Trace "RecordSet.Field.Count=" & lngCount & vbNewLine
If (lngCount>0) Then
DBRecordset.movefirst
lngValue = DBRecordset.Fields(0).Value
HMIRuntime.Trace "RecordSet.Fields(0).Value=" & lngValue & " (ValueID)"
& vbNewLine
lngValue = DBRecordset.Fields(1).Value
HMIRuntime.Trace "RecordSet.Fields(1).Value=" & lngValue & " (TimeStamp)"
& vbNewLine
lngValue = DBRecordset.Fields(2).Value
HMIRuntime.Trace "RecordSet.Fields(2).Value=" & lngValue & " (Value)"
& vbNewLine
lngValue = DBRecordset.Fields(3).Value
HMIRuntime.Trace "RecordSet.Fields(3).Value=" & lngValue & " (QualityCode)"
& vbNewLine
lngValue = DBRecordset.Fields(4).Value
HMIRuntime.Trace "RecordSet.Fields(4).Value=" & lngValue & " (Flags)"
& vbNewLine
Dim DBValueID
Dim DBTimeStamp
Dim DBValue
Dim DBQualityCode
Dim DBFlags
Dim Count
Count = 0
HMIRuntime.Trace "before While DBReader (MaxRows=" & MaxRows & ")" & vbNewLine

' 3.1 Fetch subsequent rows of the result set
DBRecordset.movefirst
Do While (Not DBRecordset.EOF And Count < MaxRows)
  DBValueID = DBRecordset.Fields(0).Value
  DBTimeStamp = DBRecordset.Fields(1).Value
  DBValue = DBRecordset.Fields(2).Value
  DBQualityCode = DBRecordset.Fields(3).Value
  DBFlags = DBRecordset.Fields(4).Value
  Count = Count + 1
  DBRecordset.movenext
Loop
HMIRuntime.Trace Count & "records read finish" & vbNewLine
Else
  HMIRuntime.Trace "Selection returned no fields" & vbNewLine
End If

' 4.1 Clean up
DBRecordset.Close
Set DBRecordset = Nothing
Set DBCommand = Nothing
DBConnection.Close
Set DBConnection = Nothing
1.15.2.14 Dynamization of Controls

Example: Calling Methods of an ActiveX Control

Introduction
The following examples illustrate how to call methods and properties of an ActiveX control which is embedded in a WinCC picture.

Example 1: WinCC FunctionTrendControl

This example fills "Trend 1" of the FunctionTrendControl "Control1" with values which describe a parabola.

To dynamize a trend with VBS, in the configuration dialog of the control on the "Data connection" tab under "Data supply" set "0 - None".

```vbscript
'VBS300
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Dim objTrend

Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")

For lngFactor = -100 To 100
    dblAxisX = CDbl(lngFactor * 0.02)
    dblAxisY = CDbl(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
    objTrend.InsertData dblAxisX, dblAxisY
Next
```

Example 2: WinCC FunctionTrendControl with value supply via array

In this example, "Trend 1" of the FunctionTrendControl "Control1" is supplied with values stored in arrays.
To dynamize a trend with VBS, in the configuration dialog of the control on the "Data connection" tab under "Data supply" set "0 - None".

```vbs
'VBS301
Dim lngIndex
Dim dblAxisX(100)
Dim dblAxisY(100)
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
For lngIndex = 0 To 100
    dblAxisX(lngIndex) = CDbl(lngIndex * 0.8)
    dblAxisY(lngIndex) = CDbl(lngIndex)
Next
objTrend.InsertData dblAxisX, dblAxisY
```

**Example 3: WinCC FunctionTrendControl (before WinCC V7)**
This example fills the FunctionTrendControl named "Control1" with values that describe a parabola.

```vbs
'VBS111
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Set objTrendControl = ScreenItems("Control1")
For lngFactor = -100 To 100
    dblAxisX = CDbl(lngFactor * 0.02)
    dblAxisY = CDbl(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
    objTrendControl.DataX = dblAxisX
    objTrendControl.DataY = dblAxisY
    objTrendControl.InsertData = True
Next
```
Example 4: WinCC FunctionTrendControl with value supply via array (before WinCC V7)

In this example, a FunctionTrendControl called "Control1" is supplied with 100 value pairs. In order that the value pair can be transferred correctly, the transfer e.g. in "dblAxisXY" must not occur directly but via an intermediate tag, e.g. "varTemp".

```vbs
'VBS152
Dim lngIndex
Dim dblXY(1)
Dim dblAxisXY(100)
Dim varTemp
Dim objTrendControl
Set objTrendControl = ScreenItems("Control1")
For lngIndex = 0 To 100
    dblXY(0) = CDbl(lngIndex * 0.8)
    dblXY(1) = CDbl(lngIndex)
    dblAxisXY(lngIndex) = dblXY
Next
varTemp = (dblAxisXY)
objTrendControl.DataXY = varTemp
objTrendControl.InsertData = True
```

Example 5: Microsoft Web Browser

This example controls MS Web Browser.

```vbs
'VBS112
Dim objWebBrowser
Set objWebBrowser = ScreenItems("WebControl")
objWebBrowser.Navigate "http://www.siemens.de"

... objWebBrowser.GoBack
... objWebBrowser.GoForward
... objWebBrowser.Refresh
... objWebBrowser.GoHome
... objWebBrowser.GoSearch
... objWebBrowser.Stop
```

Note

Insert the instructions, separated by stops, in self-defined procedures. Declaration and assignments must always precede them.
Example: How to configure a user-defined toolbar button with a self-created selection dialog

Introduction

In the following example you create a user-defined toolbar button of an OnlineTrendControl. On this toolbar button you configure a self-created selection dialog with which you can optionally set one of two different time ranges of the OnlineTrendControl.

Requirement

- The Graphics Designer is open.
- An archive is created in the Tag Logging Editor.

Inserting and configuring WinCC OnlineTrendControl

2. Save the process picture under "OnlineTrend.pdl".
3. Insert a WinCC OnlineTrendControl into the process picture.
4. Select "Configuration dialog..." from the shortcut menu of the control.
   The "Properties of WinCC OnlineTrendControl" dialog opens.
5. On the "Trend" tab under "Data connection" connect the trend to an archive tag.
6. On the "Toolbar" tab under "Button functions" create a new user-defined toolbar button with object ID "1001" for the OnlineTrendControl.
7. Click on "Accept" to save the changes.
8. Click "OK" to close the dialog box.
9. Select "Properties" from the shortcut menu of the control.
   The "Object properties" dialog box opens.
10. Enter "Control1" as the object name for the control.
11. In the Object Properties of "Control1" select the "Event" tab.
12. On the "OnToolbarButtonClick" object event, configure the VB script "Create VBS action on "OnToolbarButtonClick" event of user-defined toolbar button (VBS302)".
13. Close the "Object properties" dialog box.

Creating a process picture for the selection dialog

2. Save the process picture under "Selectiondialog.pdl".
3. Click the "Properties" button on the shortcut menu of the process picture.
   The "Object properties" dialog box opens.
5. Close the "Object properties" dialog box.
6. Insert two "Button" objects into the process picture.
7. Enter "Morning" or "Afternoon" as text for the button.

Dynamizing selection dialog button
1. In the Object Properties of the "Morning" button select the "Event" tab.
2. On the "Mouse-click" event, configure the VB script "Create VBS action on "Mouse-click"
   event of "Morning" button (VBS303)".
3. Close the "Object properties" dialog box.
4. In the Object Properties of the "Afternoon" button select the "Event" tab.
5. On the "Mouse-click" event configure the VB script "Create VBS action on "Mouse-click"
   event of "Afternoon" button (VBS304)".
6. Close the "Object properties" dialog box.

Inserting and configuring a picture window
1. Insert a "Picture window" object into the "OnlineTrend.pdl" process picture.
2. Select "Properties" from the shortcut menu of the picture window.
   The "Object properties" dialog box opens.
3. Enter "PictureWindow1" as the object name for the picture window.
4. Under "Miscellaneous" set the "Display" attribute to "no".
5. Under "Miscellaneous", select the "Selectiondialog.pdl" process picture for the "Picture
   name" attribute.
6. Close the "Object properties" dialog box.

Create VBS action on "OnToolBarButtonClick" event of user-defined toolbar button (VBS302)

'VBS302
'Open selection window if Toolbarbutton with ID 1001 is pressed
If lId = 1001 Then
  ScreenItems("PictureWindow1").Visible = True
End If
Create VBS action on "Mouse-click" event of "Morning" button (VBS303)

'VBS303
Dim obj
Set obj = Parent.Parent.ScreenItems("Control1")

'choose time axis, stop update, set begin time and time range
obj.TimeAxisName = "Time axis 1"
obj.TimeAxisActualize = False
obj.TimeAxisBeginTime = CStr(Date & " 4:00:00")
obj.TimeAxisTimeRangeBase = 3600000
obj.TimeAxisTimeRangeFactor = 8

'close the selection window
Parent.Visible = False

Create VBS action on "Mouse-click" event of "Afternoon" button (VBS304)

'VBS304
Dim obj
Set obj = Parent.Parent.ScreenItems("Control1")

'choose time axis, stop update, set begin time and time range
obj.TimeAxisName = "Time axis 1"
obj.TimeAxisActualize = False
obj.TimeAxisBeginTime = CStr(Date & " 12:00:00")
obj.TimeAxisTimeRangeBase = 3600000
obj.TimeAxisTimeRangeFactor = 8

'close the selection window
Parent.Visible = False

Example: How to add elements to an empty WinCC OnlineTrendControl

Introduction

In the following example you insert the Trend Window, Value Axis, Time Axis and Trends elements into an empty WinCC OnlineTrendControl.

Requirement

- The Graphics Designer is open.
- An archive is created in the Tag Logging Editor with three archive tags.
Inserting and configuring WinCC OnlineTrendControl

2. Insert a WinCC OnlineTrendControl into the process picture.
3. Select "Configuration dialog..." from the shortcut menu of the control.
   The "Properties of WinCC OnlineTrendControl" dialog opens.
4. In the "Trends" area of the "Trends" tab delete the default trend window "Trend 1".
5. Click on "Accept" to save the changes.
6. Click "OK" to close the dialog box.
7. Select "Properties" from the shortcut menu of the control.
   The "Object properties" dialog box opens.
8. Enter "Control1" as the object name for the control.
9. Close the "Object properties" dialog box.

Inserting and configuring a button

1. Insert a "Button" object into the process picture.
2. Enter "Paste elements" as text for the button.
3. Select "Properties" from the shortcut menu of the button.
   The "Object properties" dialog box opens.
4. In the Object Properties of the button select the "Event" tab.
5. On the "Mouse-click" event configure the VB script "Create VBS action on "Mouse-click"
   event of button (VBS305)".
6. Close the "Object properties" dialog box.
Create VBS action on "Mouse-click" event of button (VBS305)

'VBS305
Dim objTrendControl
Dim objTrendWindow
Dim objTimeAxis
Dim objValueAxis
Dim objTrend

'create reference to TrendControl
Set objTrendControl = ScreenItems("Control1")

'create reference to new window, time and value axis
Set objTrendWindow = objTrendControl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = objTrendControl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValueAxis = objTrendControl.GetValueAxisCollection.AddItem("myValueAxis")

'assign time and value axis to the window
objTimeAxis.TrendWindow = objTrendWindow.Name
objValueAxis.TrendWindow = objTrendWindow.Name

'add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "TestArchive\ArchivTag1"
objTrend.Color = RGB(255,0,0)
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name

'add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend2")
objTrend.Provider = 1
objTrend.TagName = "TestArchive\ArchivTag2"
objTrend.Color = RGB(0,255,0)
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name

'add new trend and assign properties
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myTrend3")
objTrend.Provider = 1
objTrend.TagName = "TestArchive\ArchivTag3"
objTrend.Color = RGB(0,0,255)
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name
Note
In the VB script, replace the archive used and the archive tags "Archive\ArchiveTagX" with the names of the archive and archive tags that have been created.

Example: How to add a trend and a setpoint trend to an empty OnlineTrendControl.

Introduction
In the following example, you add a trend and a setpoint trend to an empty WinCC OnlineTrendControl. The time axis and value axis are added for the trends in a trend window.

Requirement
- A "WinCC OnlineTrendControl" with the name "Control2" is inserted in the process picture in the Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured the event "mouse click", for example, for the button, with a VBS action and the following script.
Example

'VBS352
Dim objTrendControl
Dim objTrendWindow
Dim objTimeAxis
Dim objValueAxis
Dim objTrend
	'tags used to generate trend data
Dim dtCurrent
Dim dblCurrent
Dim lIndex
Dim vValues(360)
Dim vTimeStamps(360)

'create reference to TrendControl
Set objTrendControl = ScreenItems("Control2")

'---- reference trend ----
'create reference to new window, time and value axis
Set objTrendWindow = objTrendControl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = objTrendControl.GetTimeAxisCollection.AddItem("myRefTimeAxis")
Set objValueAxis = objTrendControl.GetValueAxisCollection.AddItem("myRefValueAxis")

'assign time and value axis to the window
objTimeAxis.TrendWindow = objTrendWindow.Name
objTimeAxis.ShowDate = False
objValueAxis.TrendWindow = objTrendWindow.Name

'add trend and assign propertys
Set objTrend = objTrendControl.GetTrendCollection.AddItem("myRefTrend")
oobjTrend.Provider = 0
objTrend.Color = RGB(0,0,0)
objTrend.TrendWindow = objTrendWindow.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValueAxis.Name

'generate values for reference trend
dtCurrent = CDate("23.11.2006 00:00:00")
For lIndex = 0 To 360
	vValues(lIndex) = ( Sin(dblCurrent) * 60 ) + 60
	vTimeStamps(lIndex) = dtCurrent
	dblCurrent = dblCurrent + 0.105

dtCurrent = dtCurrent + CDate ("00:00:01")
Next

'insert data to the reference trend
objTrend.RemoveData
objTrend.InsertData vValues, vTimeStamps

'---- data trend ----
'add time and value axis to the existing window
Set objTimeAxis = objTrendControl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValueAxis = objTrendControlGetValueAxisCollection.AddItem("myValueAxis")

'assign time and value axis to the window
Example: How to add elements to a WinCC OnlineTrendControl

Introduction

In the following example, insert value columns with properties in an empty WinCC
OnlineTableControl and link the columns to archive tags.
**Requirement**

- An archive is created in the “Tag Logging Editor” with three archive tags.
- A “WinCC OnlineTableControl” with the name “Control2” is inserted in the process picture in the Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured the event "mouse click", for example, for the button, with a VBS action and the following script.

**Example**

```vbscript
'VBS351
Dim objTableControl
Dim objTimeColumn
Dim objValueColumn
Dim objTrend

'create reference to TableControl and enable BackColor
Set objTableControl = ScreenItems("Control2")
objTableControl.UseColumnBackColor = True

'create reference to new TimeColumn and assign column length
Set objTimeColumn = objTableControl.GetTimeColumnCollection.AddItem("myRefTimeAxis")
objTimeColumn.Length = 20

'add new ValueColumn and assign properties
Set objValueColumn = objTableControl.GetValueColumnCollection.AddItem("myValueTable1")
objValueColumn.Provider = 1
objValueColumn.TagName = "Process value archive\PDL_ZT_1"
objValueColumn.BackColor = RGB(255,255,255)
objValueColumn.TimeColumn = objTimeColumn.Name

'add new ValueColumn and assign properties
Set objValueColumn = objTableControl.GetValueColumnCollection.AddItem("myValueTable2")
objValueColumn.Provider = 1
objValueColumn.TagName = "Process value archive\PDL_ZT_2"
objValueColumn.BackColor = RGB(0,255,255)
objValueColumn.TimeColumn = objTimeColumn.Name

'add new ValueColumn and assign properties
Set objValueColumn = objTableControl.GetValueColumnCollection.AddItem("myValueTable3")
objValueColumn.Provider = 1
objValueColumn.TagName = "Process value archive\PDL_ZT_3"
objValueColumn.BackColor = RGB(255,255,0)
objValueColumn.TimeColumn = objTimeColumn.Name
```
Example: Scripts for WinCC AlarmControl

Introduction

The following examples demonstrate the use of scripts for WinCC AlarmControl.

Requirement

- You have already configured messages in the "Alarm Logging" editor.

Example 1: Setting filters

A filter with message number "2" is set, or reset if the filter has already been set. The status is also output in the dialog window.

```
'VBS353
Dim objAlarmControl
'create reference to AlarmControl
Set objAlarmControl = ScreenItems("Controll")
' set / reset the filter and create a trace
If (objAlarmControl.MsgFilterSQL = "") Then
    objAlarmControl.MsgFilterSQL = "MSGNR = 2"
    HMIRuntime.Trace "MsgFilterSQL set to MSGNR = 2" & vbNewLine
Else
    objAlarmControl.MsgFilterSQL = ""
    HMIRuntime.Trace "no filter" & vbNewLine
End If
```
Example 2: Adding a column to WinCC AlarmContol

The column "Message text" is added or removed if the column already exists. The status is also output in the dialog window. The message block of the "Message text" column has the object name "Text1".

'VBS354
'add this function to the declaration section
Function IsExistingMsgColumn( objAlarmControl, strName )
' this function checks if the MessageColumn exists
on error resume next
objAlarmControl.GetMessageColumn( strName )
If err.number = 0 Then
   IsExistingMsgColumn = True
else
   err.Clear
   IsExistingMsgColumn = False
End if
End Function

'example code
Dim objAlarmControl
Dim colMsgColumn
'create reference to the alarm control
Set objAlarmControl = ScreenItems("Control1")
Set colMsgColumn = objAlarmControl.GetMessageColumnCollection
'add or remove the MsgColumn
If ( IsExistingMsgColumn(objAlarmControl, "Text1") ) Then
   HMIRuntime.Trace "Remove MsgColumn" & vbNewLine
   colMsgColumn.RemoveItem("Text1")
Else
   HMIRuntime.Trace "Add MsgColumn" & vbNewLine
   colMsgColumn.AddItem("Text1")
End If

Example 3: Output content of message window in dialog window

'VBS355
Dim objAlarmControl
Dim lIndex
Dim lCellIndex
'create reference to the alarm control
Set objAlarmControl = ScreenItems("Control1")
'enumerate and trace out row numbers
For lIndex = 1 To objAlarmControl.GetRowCollection.Count
   HMIRuntime.trace "Row: " & (objAlarmControl.GetRow(lIndex).RowNumber) & " "
'enumerate and trace out column titles and cell texts
   For lCellIndex = 1 To objAlarmControl.GetRow(lIndex).CellCount
      HMIRuntime.trace objAlarmControl.GetMessageColumn(lCellIndex -1).Name & " "
      HMIRuntime.trace objAlarmControl.GetRow(lIndex).CellText(lCellIndex) & " "
      Next
   Next
   HMIRuntime.trace vbNewLine
Next
1.15.3 General Examples

1.15.3.1 General examples for VBScript

Introduction

This section contains examples of the general use of VBScript with regard to the following topics:

- Program data connection with VBS
- To retrieve methods
- Using the MS Automation Interface
- Starting External Applications

Note

All objects supplied with the Windows Script Host (WSH) from Microsoft can be integrated in their environment using the standard VBS method "CreateObject". However, there is no direct access to the WSH object itself using VBS from WinCC.

Example 1: "FileSystemObject" object for working with the file system

Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("This is a test.")
MyFile.Close

Example 2: "WScript.Shell"-Object for working with the Windows environment

See also

Example: Configuring a Database Connection with VBS (Page 856)
Example: Starting an external application (Page 859)
Example: Using the MS Automation interface (Page 858)
Example: Calling Methods of an ActiveX Control (Page 841)
1.15.3.2 Example: Configuring a Database Connection with VBS

Introduction

The following examples describe the configuration of an Access database link via an ODBC driver.

- Example 1 writes a tag value from WinCC in an Access database.
- Example 2 reads a value from the database and writes it in a WinCC tag.

The examples do not contain any handling faults.

Procedure, Example 1

1. Create the Access database with the WINCC_DATA table and columns (ID, TagValue) with the ID as the Auto Value.
2. Set up the ODBC data source with the name "SampleDSN", reference to the above Access database.

Example 1

'VBS108
Dim objConnection
Dim strConnectionString
Dim lngValue
Dim strSQL
Dim objCommand
strConnectionString = "Provider=MSDASQL;DSN=SampleDSN;UID=;PWD=;"
lngValue = HMIRuntime.Tags("Tag1").Read
strSQL = "INSERT INTO WINCC_DATA (TagValue) VALUES (" & lngValue & ");"
Set objConnection = CreateObject("ADODB.Connection")
objConnection.ConnectionString = strConnectionString
objConnection.Open
Set objCommand = CreateObject("ADODB.Command")
With objCommand
  .ActiveConnection = objConnection
  .CommandText = strSQL
End With
objCommand.Execute
Set objCommand = Nothing
objConnection.Close
Set objConnection = Nothing
Procedure, Example 2

1. Create the WinCC tag with the name dbValue.
2. Create Access database with WINCC_DATA table and ID, TagValue columns: ID, create TagValue (ID as auto value).
3. Set up the ODBC data source with the name "SampleDSN", reference to the above Access database.

Example 2

'VBS108a
Dim objConnection
Dim objCommand
Dim objRecordset
Dim strConnectionString
Dim strSQL
Dim lngValue
Dim lngCount
strConnectionString = "Provider=MSDASQL;DSN=SampleDSN;UID=;PWD=;"
strSQL = "select TagValue from WINCC_DATA where ID = 1"
Set objConnection = CreateObject("ADODB.Connection")
Set objRecordset = CreateObject("ADODB.Recordset")
Set objCommand = CreateObject("ADODB.Command")
objCommand.ActiveConnection = objConnection
objCommand.CommandText = strSQL
Set objRecordset = objCommand.Execute
lngCount = objRecordset.Fields.Count
If lngCount>0 Then
    lngValue = objRecordset.Fields(0).Value
    HMIRuntime.Tags("dbValue").Write lngValue
Else
    HMIRuntime.Trace "Selection returned no fields" & vbNewLine
End If
Set objCommand = Nothing
objConnection.Close
Set objRecordset = Nothing
Set objConnection = Nothing

There are several ways in which to define the ConnectionString for the connection depending on the provider used:
Microsoft OLE DB provider for ODBC

Enables connections to any ODBC data source. The corresponding syntax is:

```
"[Provider=MSDASQL;]{DSN=name|FileDSN=filename};
[DATABASE=database;]UID=user; PWD=password"
```

Other Microsoft OLE DB Providers (e.g. MS Jet, MS SQL Server)

It is possible to work without DSN. The corresponding syntax is:

```
"[Provider=provider;]DRIVER=driver; SERVER=server;
DATABASE=database; UID=user; PWD=password"
```

See also

General examples for VBScript (Page 855)

1.15.3.3 Example: Using the MS Automation interface

Introduction

The following three examples illustrate how to use the MS Automation interface.

Example 1: MS Excel

In this example, an output value from an input field is written in an Excel table.

```
'VBS113
Dim objExcelApp
Set objExcelApp = CreateObject("Excel.Application")
objExcelApp.Visible = True
'
'ExcelExample.xls is to create before executing this procedure.
'Replace <path> with the real path of the file ExcelExample.xls.
objExcelApp.Workbooks.Open "<path>\ExcelExample.xls"
objExcelApp.Cells(4, 3).Value = ScreenItems("IOField1").OutputValue
objExcelApp.ActiveWorkbook.Save
objExcelApp.Workbooks.Close
objExcelApp.Quit
Set objExcelApp = Nothing
```
Example 2: MS Access

This example opens a report from MS Access.

'VBS114
Dim objAccessApp
Set objAccessApp = CreateObject("Access.Application")
objAccessApp.Visible = True
' 'DbSample.mdb and RPT_WINCC_DATA have to create before executing 'this procedure.
'Replace <path> with the real path of the database DbSample.mdb.
objAccessApp.OpenCurrentDatabase "<path>\DbSample.mdb", False
objAccessApp.CloseCurrentDatabase
Set objAccessApp = Nothing

Example 3: MS Internet Explorer

This example opens the MS IE.

'VBS115
Dim objIE
Set objIE = CreateObject("InternetExplorer.Application")
objIE.Navigate "http://www.siemens.de"
Do
Loop While objIE.Busy
objIE.Resizable = True
objIE.Width = 500
objIE.Height = 500
objIE.Left = 0
objIE.Top = 0
objIE.Visible = True

See also

General examples for VBScript (Page 855)

1.15.3.4 Example: Starting an external application

Introduction

The following two examples illustrate how to start an external application.
Example:

'VBS117
Dim objWshShell
Set objWshShell = CreateObject("Wscript.Shell")
objWshShell.Run "Notepad Example.txt", 1

See also

General examples for VBScript (Page 855)
1.16 Basic Principles of VBScript

1.16.1 Basic Principles of VBScript

Introduction

The most important topics of the Microsoft VBScript Reference are provided below:

- VBScript Language Directory
- VBScript Tutorial with the most important basic principles
- Scripting runtime reference

If a full version of the VBScript Reference is required, it is available under http://msdn2.microsoft.com/en-us/library/t0aew7h6 (http://msdn2.microsoft.com/en-us/library/t0aew7h6)

See also

Microsoft VBScript Reference (http://msdn2.microsoft.com/en-us/library/t0aew7h6)

1.16.2 VBScript Basics
1.16 Basic Principles of VBScript
2.1 Creating Functions and Actions with ANSI-C

Contents

In Runtime, background tasks, such as printing daily reports, monitoring tags or performing picture-specific calculations, are performed as actions.

These actions are started by triggers.

Functions can be called from actions. WinCC has a multitude of functions, which can be modified by the user. Furthermore, the user can also develop his own functions.

The Global Script editor is used to create and edit functions and actions.

This chapter will show you

● How to use the Global Script editor
● How to create and edit functions
● How to create and edit actions
● How to use the diagnostic tools to analyze runtime problems
2.2 Creating Functions and Actions

Introduction

WinCC supports the use of functions and actions for dynamization of the processes in your WinCC project. These functions and actions are written in ANSI-C.

Difference between Functions and Actions

Actions are activated by a trigger, namely a triggering event. Functions do not have a trigger and are used as components of actions as well as in Dynamic Dialogs, in Tag Logging and in Alarm Logging.

Trigger Types

The following trigger types are available:
Outline of the Functions and Actions

The diagram provides an overview of the range of functions and actions:

![Functions and Actions Diagram]

Actions are used for picture-independent background tasks, such as printing daily reports, monitoring tags or performing calculations.

Functions are pieces of code, which can be used in several locations, but are only defined in one place. WinCC includes a multitude of functions. Furthermore, you can also write your own functions and actions.

The included standard functions can be modified by the user. In the event that WinCC is reinstalled or upgraded, the standard functions that were modified are deleted or replaced by the unedited standard functions. Therefore, you should back up the modified functions prior to upgrading or reinstalling.

Design tool

WinCC provides the "Global Script" editor for designing, creating and editing functions and actions. Global Script is started from the navigation window of WinCC Explorer.

Unicode support

You can set the suitable code page in the toolbar of the "Global Script" editor. This means that the system language no longer has to be changed with the Microsoft setting "Start > Settings > Control Panel > Regional and Language Options".

You can select "Dynamic: Project setting" as the language setting for scripts. The C script is compiled in English. The code page of the centrally configured language is used for the strings in runtime.
You can specify the project setting in the "Project Properties" dialog in the WinCC Explorer. You can select the following from a list in the "Options" under "C scripts with "Dynamic" language setting in runtime":

- "Respective set WinCC Runtime language". The C script is executed in the WinCC Runtime language.
- Operating system language for non-Unicode programs.
- The C script is executed with the code page setting of the operating system. Select the language from the list.

See also

Runtime Behavior of Actions (Page 927)
How To Create and Edit Actions (Page 905)
Creating and Editing Functions (Page 892)
The Global Script Editor (Page 878)
Use of DLLs in Functions and Actions (Page 876)
Use of Global C-Tags (Page 874)
How to Add Global Script Runtime to a Project's Startup List (Page 873)
How to Generate a New Header (Page 887)
Characteristics of Global Actions (Page 872)
Characteristics of Local Actions (Page 871)
Characteristics of Internal Functions (Page 870)
Characteristics of Standard Functions (Page 868)
Characteristics of Project Functions (Page 867)
2.3 Characteristics of Project Functions

Characteristics of Project Functions

- can be created by yourself
- can be edited by you
- can be password-protected against modification and viewing by unauthorized persons
- have no trigger
- are only known within the project
- are assigned file name extension ".fct"

Project functions are saved in the "\library" subdirectory of the WinCC project.

Use of Project Functions

- in other project functions
- in Global Script actions
- in C-Actions in the Graphics Designer and within the Dynamic Dialog
- in Alarm Logging within the Loop in Alarm functionality
- in Tag Logging when starting and releasing archives and when swapping-out cyclic archives

See also

How to Protect a Function Against Unauthorized Access (Page 900)
Creating and Editing Functions (Page 892)
2.4 Characteristics of Standard Functions

Characteristics of Standard Functions

Standard functions ...  
- are provided for use in WinCC  
- cannot be created by yourself  
- can be edited by you  
- can be password-protected against modification and viewing by unauthorized persons  
- have no trigger  
- are known across projects  
- are assigned file name extension "*.fct"

Standard functions are saved in the "aplib" subdirectories in the WinCC installation directory.

Use of Standard Functions

Standard functions can be used...  
- in project functions  
- in other standard functions  
- in Global Script actions  
- in C-actions in the Graphics Designer and within the Dynamic Dialog  
- in Alarm Logging within the Loop in Alarm functionality  
- in Tag Logging when starting and releasing archives and when swapping-out cyclic archives
2.4 Characteristics of Standard Functions

Note
The included standard functions can be edited by the user. In the event that WinCC is reinstalled or upgraded, the standard functions that were modified are deleted or replaced by the unedited standard functions. Therefore, you should back up the modified functions prior to upgrading or reinstalling.

See also
How to Use Standard and Project Functions (Page 897)
Creating and Editing Functions (Page 892)
2.5 Characteristics of Internal Functions

Characteristics of Internal Functions

Internal functions ...
- are provided for use in WinCC
- cannot be created by you
- cannot be edited
- cannot be renamed
- have no trigger
- are know project-wide
- are assigned file name extension "*.icf"

Internal functions are saved in the "apllib" subdirectories in the WinCC installation directory.

Use of Internal Functions

Internal functions can be used...
- in project functions
- in standard functions
- in actions
- in C-actions in the Graphics Designer and within the Dynamic Dialog
2.6 Characteristics of Local Actions

Characteristics of Local Actions
Local actions ...

- can be created by yourself
- can be edited by you
- can be password-protected against modification and viewing by unauthorized persons
- have at least one trigger
- are only executed on the assigned computer
- are assigned file name extension "*.pas"

Local actions are saved in the "\<computer_name\>\Pas" subdirectory in the project directory.

Use of Local Actions
Actions are used for picture-independent background tasks, such as printing daily reports, monitoring tags or performing calculations. An action is started by the trigger configured for it. In order for an action to be executed, Global Script Runtime must be included in the startup list.

In contrast to global actions, local actions can be assigned to a single computer. It is thus for example possible to ensure that a report is only printed on the server.

See also
How to Protect an Action Against Unauthorized Access (Page 912)
Triggers (Page 914)
How To Create and Edit Actions (Page 905)
How to Add Global Script Runtime to a Project's Startup List (Page 873)
2.7 Characteristics of Global Actions

Characteristics of Global Actions

Global actions ...

- can be created by yourself
- can be edited by you
- can be password-protected against modification and viewing by unauthorized persons
- have at least one trigger to start them
- are executed on all project computers in a client-server project
- are assigned file name extension "*.pas"

Global Actions are saved in the "\Pas" subdirectory of the WinCC project.

Use of Global Actions

Actions are used for background tasks, such as printing daily reports, monitoring tags or performing calculations. An action is started by the trigger configured for it. In order for an action to be executed, Global Script Runtime must be included in the startup list.

In contrast to local actions, global actions are executed on all project computers in a client-server project. In a single-user project there is no difference between global and local actions.

See also

- How to Protect an Action Against Unauthorized Access (Page 912)
- Triggers (Page 914)
- How To Create and Edit Actions (Page 905)
- How to Add Global Script Runtime to a Project's Startup List (Page 873)
2.8 How to Add Global Script Runtime to a Project's Startup List

Introduction

In order to run Global Script Actions in Runtime, Global Script Runtime must be added to the project's startup list. This does not affect the executability of the functions.

Procedure

1. In the shortcut menu of computer in WinCC Explorer, select "Properties". The "Computer list properties" dialog opens.
2. Click "Properties". The "Computer Properties" dialog opens.
3. Select the "Startup" tab
4. Activate "Global Script Runtime".
5. Click "OK" to close the dialog.
2.9 Use of Global C-Tags

Definition of global C tags

A global C-tag is defined by adding the definition line in front of the function name of a function:

```c
int a; // The tag a is defined as an integer
global void dummy() // Function name
{
  // Function code
}
```

Validity range

A tag defined in this manner is known to every function and action in Runtime. It is created as soon as Runtime is started, even if the function itself was not called.

Note

When you operate the WinCC Service Mode, there is no common data area for C scripting. Thus, for example, no global C variables can be exchanged between "Global Script" and the "Graphics Designer".

Use of global C tags

Global C tags are used in functions or actions by declaring them as external within the function or action:

```c
void dummy() // Function name
{
  extern int a; // External declaration of the tag a
  // Function code
}
```

The compiler is thus informed that it need not create the tag, since it is created in another location in Runtime.

If the value of tag a changes, this change can be read by every function and action.

Each C-tag may only be defined in one location. For reasons of clarity and to avoid duplicate definitions, we recommend defining global C tags in only one location.
Note

A maximum of 64 Kbytes are available to a function and the global C-tag defined with it.
2.10 Use of DLLs in Functions and Actions

Adjusting DLLs

WinCC allows you to use your own DLLs (Dynamic Link Libraries).

Functions in existing DLLs can be enabled for functions and actions by making the necessary additions to the respective function or action.

Add the following code in front of the function or action:

```c
#pragma code("<Name>.dll")
<Type of returned value> <Function_name 1>(...);
<Type of returned value> <Function_name2>(...);
.
.
.
<Type of returned value> <Function_name n>(...);
#pragma code()
```

The functions `<Function_name 1>` ... `<Function_name n>` from `<Name.dll>` are declared and can now be called by the respective function or action.

Example:

```c
#pragma code("kernel32.dll")
VOID GetLocalTime(LPSYSTEMTIME lpSystemTime);
#pragma code()

SYSTEMTIME st;

GetLocalTime(&st);
```

As an alternative to this procedure, you can also make the necessary additions in the "Apdefap.h" header file.

When using own DLLs in WinCC, you must use the release version. WinCC is delivered as a release version and thus uses the release version of the system DLLs. If you generate a custom DLL in the debug version, it is possible that both the release and the debug version of the DLL are loaded, increasing the memory requirements.
Structures of the DLL have to be set up using 1-byte alignment.

**Note**

The DLL must be saved in either the "bin" directory or in a path defined in the "PATH" system tag. This tag is defined in the operating system properties.
2.11 The Global Script Editor

2.11.1 The Global Script Editor

Introduction

WinCC supports the creation and editing of functions and actions with the Global Script editor. Global Script is started from the project window of WinCC Explorer.

Structure of the Global Script Editor

The Global Script editor is designed in accordance with the Windows standards. It comes with toolbars, a menu bar and a status bar. It has several windows featuring drop-down menus.

Menu bar (1)
The menu bar content depends on the situation. It is always visible.

Toolbars (2)
Global Script contains two toolbars. You may always unhide these toolbars and use the mouse.
to move them to any screen position. You may hide/unhide the toolbars using the "View" > "Toolbars" menu command and move these to any position in the editor.

Navigation window (3)
The navigation window serves to select functions and actions for editing, or to insert an editing window at the cursor position. The functions and actions are organized in groups with hierarchic order. Functions are always displayed by their function name, while actions are displayed by their file name.

Editing window (4)
The editing window is used to edit functions and actions. It is only visible when a function or action has been opened for editing. Each function or action is opened in a separate edit window. Several editing windows can be opened simultaneously.

Output window (5)
The output window displays results of the "Search in files" or "Compile all functions" functions. By default, it is visible, but can be hidden.

- Search in files:
  A hit list is returned for each search term found in a single line of the output window, which consists of the line number, path and file name, as well as the line text with the specified number of the line in which the search term was found. You can directly open the corresponding file by double-clicking an entry in the output window. The cursor is placed in the line in which the search term was found.

- Compile All Functions:
  Compiler warnings and error messages are output for all functions compiled. In the next line, the path and file name of the compiled function as well as the summary message from the compiler are displayed.

Status bar (6)
The status bar is located on the bottom edge of the Global Script window; you may hide/unhide the status bar. It contains information about the position of the cursor in the edit window and the keyboard settings. In addition, the status bar shows either a brief description for the currently selected Global Script functionality or a tip.

Window docking

Window docking is a useful tool for the flexible arrangement of windows. It lets you reposition windows to obtain separate windows, or to group windows in tab groups. For example, you can arrange your actions horizontally, vertically, or as tab group. You may hide windows automatically and show these again as required. For more information, refer to chapter "Creating process pictures".

See also

- Printing Functions and Actions (Page 890)
- How to Search in Files (Page 889)
- How to Compile All Functions (Page 888)
- How to Generate a New Header (Page 887)
2.11.2 Working in the Edit Window

2.11.2.1 Working in the Edit Window

Introduction

The edit window includes an array of functions, which can be executed with either the keyboard or the mouse.

In the edit window, you can edit functions and actions.

```c
#include "apdefap.h"

int gscAction( void )
{
    //WINCC:TAGNAME SECTION_START
    //syntax: #define TagNameInAction "DMTagName"
    //next TagID = 1
    //WINCC:TAGNAME SECTION_END

    //WINCC:PICNAME SECTION_START
    //syntax: #define PicNameInAction "PictureName"
    //next PicID = 1
    //WINCC:PICNAME SECTION_END

    return 0;
}
```

Compiling...
0 Error(s), 0 Warning(s)

Contents

The window can be split. The upper part of the edit window displays the code of the function or action. In the lower part, you can see the messages that the compiler output while compiling the function or action.
Properties

When the window is opened for the first time, the lower part of the window is minimized. When the compiler process is started, the lower part of the window is enlarged to allow for the display of the compiler messages. The division of the window can be adjusted with the mouse. Double-click an error message to jump to the corresponding line in the code.

Color code

The C code is color-coded as follows:

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Keywords</td>
<td>define, double, if</td>
</tr>
<tr>
<td>Green</td>
<td>Comments</td>
<td>// is a comment</td>
</tr>
<tr>
<td>Red</td>
<td>Strings</td>
<td>&quot;Rectangle3&quot;</td>
</tr>
<tr>
<td>Black</td>
<td>Other C-codes</td>
<td>level=100*newvalue/255;</td>
</tr>
</tbody>
</table>

Note

A function or action cannot have more than 32767 characters including spaces.

See also

Editing Functions with the Mouse (Page 882)
Editing Functions with the Keyboard (Page 881)

2.11.2.2 Editing Functions with the Keyboard

You can carry out the following editing functions using the keyboard:

<table>
<thead>
<tr>
<th>Editing function</th>
<th>Keyboard operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch between write modes Insert/Overwrite</td>
<td>&lt;INSERT&gt;</td>
</tr>
<tr>
<td>Add new line</td>
<td>&lt;ENTER&gt;</td>
</tr>
<tr>
<td>Delete one character to the right</td>
<td>&lt;DELETE&gt;</td>
</tr>
<tr>
<td>Delete one character to the left</td>
<td>&lt;BACKSPACE&gt;</td>
</tr>
<tr>
<td>Delete marked text</td>
<td>&lt;DELETE&gt; or &lt;BACKSPACE&gt;</td>
</tr>
<tr>
<td>Jump to beginning of line</td>
<td>&lt;POS1&gt;</td>
</tr>
<tr>
<td>Jump to end of line</td>
<td>&lt;END&gt;</td>
</tr>
<tr>
<td>Jump to beginning of text</td>
<td>&lt;CTRL+POS1&gt;</td>
</tr>
<tr>
<td>Jump to end of text</td>
<td>&lt;CTRL+END&gt;</td>
</tr>
<tr>
<td>Move cursor</td>
<td>&lt;Cursor keys&gt;</td>
</tr>
<tr>
<td>Move cursor by one window content to beginning of text</td>
<td>&lt;PAGE UP&gt;</td>
</tr>
<tr>
<td>Move cursor by one window content to end of text</td>
<td>&lt;PAGE DOWN&gt;</td>
</tr>
<tr>
<td>Move cursor to first line in window</td>
<td>&lt;CTRL+PAGE UP&gt;</td>
</tr>
</tbody>
</table>
# Editing Functions with the Mouse

You can carry out the following editing functions using the mouse:

<table>
<thead>
<tr>
<th>Editing function</th>
<th>Mouse command (left mouse button)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select text</td>
<td>Drag mouse over text</td>
</tr>
<tr>
<td>Select a word</td>
<td>Double-click the word</td>
</tr>
<tr>
<td>Select a line</td>
<td>Triple-click the line</td>
</tr>
<tr>
<td>Extended selection</td>
<td>(&lt;\text{SHIFT}&gt; + \text{mouse-click})</td>
</tr>
<tr>
<td>Set cursor</td>
<td>Click</td>
</tr>
<tr>
<td>Move selected text</td>
<td>Drag and drop</td>
</tr>
<tr>
<td>Duplicate selected text</td>
<td>(&lt;\text{CTRL}&gt; + \text{drag and drop})</td>
</tr>
</tbody>
</table>

Other editing functions:
- By double-clicking a compiler error message, the editor jumps to the corresponding line in the code.
- Right-clicking calls up a shortcut menu.

With the following actions, the selected text is replaced by the result of the action:
- Input of character(s) through the keyboard
- Pasting of text from the clipboard
- Insertion of a function call by means of parameter assignment

## Working with the Toolbars

### Purpose

The toolbars are located in their default position below the menu bar, at the top of the Global Script window. The toolbar buttons allow for fast and easy execution to a number of Global Script functions.

There are two toolbars available:

"Standard" toolbar
"Edit" toolbar

The standard toolbar contains buttons for the following functions:

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
<th>Key combination</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Creates a new action.</td>
<td>&lt;ALT+A&gt; or &lt;CTRL+N&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Creates a new standard function.</td>
<td>&lt;ALT+S&gt; or &lt;CTRL+N&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Creates a new project function.</td>
<td>&lt;ALT+P&gt; or &lt;CTRL+N&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Creates a new header file.</td>
<td>&lt;CTRL+H&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Opens an existing action or function.</td>
<td>&lt;CTRL+O&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Saves the content of the active editing window.</td>
<td>&lt;CTRL+S&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Cuts the selected text and copies it to the clipboard.</td>
<td>&lt;CTRL+X&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Copies the selected text to the clipboard.</td>
<td>&lt;CTRL+C&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Pastes the contents of the clipboard at the location of the cursor. This function is only available if the clipboard is not empty.</td>
<td>&lt;CTRL+V&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Undoes the last of a maximum of 30 editor actions.</td>
<td>&lt;CTRL+Z&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Redoes the last editor action that was undone.</td>
<td>&lt;CTRL+A&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Prints the contents of the active editing window as project documentation. This function is only available if an editing window is open.</td>
<td>&lt;CTRL+P&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Activates the direct help (What's this?).</td>
<td>&lt;SHIFT+F1&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Generates a new header.</td>
<td>&lt;CTRL+G&gt;</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Allows you to set the font.</td>
<td>&lt;CTRL+F&gt;</td>
</tr>
</tbody>
</table>
The key combination <CTRL+N> is only available if at least one editing window is open. If the active editing window contains a function and <CTRL+N> is pressed, a new project function is created. If the active editing window contains an action, this key combination creates a new global action.

The editing toolbar contains buttons for the following functions:

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
<th>Key combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>Enables you to add information about functions and, in the case of an action, to setup a trigger. This function is only available if an editing window is open.</td>
<td>&lt;CTRL+I&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Compiles the code in the active editing window. This function is only available if an editing window is open.</td>
<td>&lt;SHIFT+F8&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Set the appropriate code page. Verify that the code page selection matches the source text. You cannot use more than one language in the source text.</td>
<td>-</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Opens the tag dialog. This function is only available if an editing window is open.</td>
<td>&lt;CTRL+R&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Opens a dialog for the selection of a picture. The name of the selected picture is inserted at the position of the cursor in the editing window. This function is only available if an editing window is open.</td>
<td>&lt;CTRL+W&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Imports an action. This function is only available if the active window contains an action.</td>
<td>&lt;CTRL+M&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Exports the action from the active editing window. This function is only available if the active window contains an action.</td>
<td>&lt;CTRL+T&gt;</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Sets the authorization for operating the action. This function is only available if the active window contains an action.</td>
<td>&lt;CTRL+E&gt;</td>
</tr>
</tbody>
</table>

Properties

Both toolbars can be shown or hidden.
They can be pinned below the menu bar. When they are not pinned down, they can be dragged with the mouse to any position on the screen.

See also
How to Set Different Views (Page 885)

2.11.4 How to Set Different Views

Introduction
In this context, views are considered to be different combinations of elements visible in the Global Script editor, such as the output window, status bar and toolbars. These elements can be individually displayed or hidden.
By default, all elements are visible.

Procedure
1. Opens the "View" menu in the Global Script menu bar.
2. Activate or deactivate the display of the desired elements e.g. the toolbars. If "show" is chosen, a check mark is displayed in front of the name.

Note
When Global Script restarted, the editor reverts to the default settings and all elements are again visible.

2.11.5 How to Set the Font Style

Introduction
The font style is composed of the settings "Font", "Style" and "Size".
The style selected is active in all edit windows.

Procedure
1. Click the button in the standard toolbar to open the dialog for setting the font style.
2. Make the desired settings.
3. Confirm your settings by clicking "OK".
Alternative procedure

You can also open the dialog for the font style settings in the following manner:

Select the "Options" menu in the Global Scripts menu bar and select "Font", or use the corresponding key combination.

---

Note

The settings are automatically saved and are not reset when WinCC is restarted.

2.11.6 How to Use "Save As..."

Introduction

If a function or action is created, Global Script saves the corresponding file in a predefined path with a default file name, e.g. "new_function_1.fct" for functions and "gsccs1.pas" for actions. Since these default file names are not particularly useful, use "Save As ..." to save the function or action under a different - more meaningful - file name. The file with the default file name is retained.

With "Save As ...", only the file name is changed, the function or action name remains unchanged.

Global Script expects that the function or action is saved in a project directory. If this is not the case, a message is displayed, but the file is saved nonetheless.

Requirement

"Save As..." is only available, if at least one edit window is open. It saves the content of the active edit window.

Procedure

1. In the Global Script menu bar, open the "File" menu.
2. Select "Save As ...".
3. Enter the new file name.
4. Close the dialog by clicking the "Save" button.

See also

How to Delete Actions or Project and Standard Functions (Page 887)
2.11.7 How to Delete Actions or Project and Standard Functions

Introduction

Actions or project and standard functions can be deleted during configuration or in Runtime. Global Script deletes the entry in the navigation window as well as the associated file.

If a deleted function is called by an action, the action is terminated upon calling the function.

If a Global Script diagnostic window is open at this time, a message is displayed. The termination of the action is logged in the "WinCC_Sys_xx.log" diagnostic file (xx = consecutive number). This diagnostic file is located in the "Diagnostics" subdirectory of the WinCC installation directory.

Procedure

1. In the Global Script navigation window, call up the shortcut menu for the function or action to be deleted.
2. Select "Delete".
3. Confirm the command by clicking "Yes".

Alternative operation

Instead of using the shortcut menu, you can also delete the selected function or action by using the <DELETE> key.

Note

If a function is deleted, the entry in the respective header file is deleted as well.

2.11.8 How to Generate a New Header

Introduction

The header must be generated again in the following cases:

- After you have copied project functions from a different project to the "library" directory in your project path.
- After you have copied standard functions from another PC to the "aplib" directory or subdirectories.

By regenerating the header, you enter the copied functions in the respective header files. You can then use the functions in your project.

Procedure

1. Click in the "Standard" toolbar.
Alternative operation

Alternatively, you can start a generation process as follows:
Open the "Options" menu and select "Regenerate Header" or use the corresponding key combination.

Note

Once the regeneration is finished, the contents of the navigation window are updated.
If WinCC is in Runtime, the Runtime system is not influenced by the regeneration of the header.

2.11.9 How to Compile All Functions

Introduction

If you have changed the header files manually, you have to recompile all functions. All project functions, standard functions and internal functions are automatically compiled with the menu command "Compile all functions".
If functions are called in other functions, error messages are possible. The reason for this is that the called functions have not yet been compiled. These functions must then be compiled individually.

Requirement

This function is only available, if all edit windows are closed.

Procedure

1. Open the "Options" menu.
2. Select "Compile All Functions".

Alternative operation

You can compile all functions using the key combination <ALT+U>.
Result

The results of the individual compilation runs are displayed in the output window, e.g. warnings and error messages of the compiler. The path and the file name of the compiled function as well as the summary message of the compiler are also displayed.

Note

In a multi-user project, the "Compile All Functions" function is not available. Assigning functions is no longer possible with these projects.

The functions compiled in this way will not become active until the next time you start Runtime on a WinCC PC.

2.11.10 How to Search in Files

Introduction

All files of the group selected in the navigation window are searched for the specified search term.

The result of the search is displayed in the output window as follows:

For each found search term, a line is displayed in the output window. This line contains the line number of the line in the code in which the search term was found, plus the path and file name as well as the line of code itself.

Standard and project functions as well as actions can be opened by double-clicking the search results. The cursor is positioned at the start of the line in which the search term was found. In the case of internal functions, the function containing the search term is shown in the navigation window and selected.

Procedure

1. Open the shortcut menu for the group to be searched in the Global Script navigation window.
2. Select "Find in Files".
3. In the dialog, enter the search term to be found.
4. Click "Find" to start the search. The result of the search is displayed in the output window.
2.11.11 Printing Functions and Actions

2.11.11.1 Printing Functions and Actions

Introduction

Actions or project or standard functions can be printed using specified system layouts. This is however only possible, if the function or action to be printed is displayed in the edit window. The content of the active edit window is printed. The printout can be examined on the screen in page view. The printing process can be controlled by a number of print parameters.

The following system layouts are used:

- @gsc_pfc.rpl for project functions
- @gsc_sfc.rpl for standard functions
- @gsc_act.rpl for actions

See also

How to Print the Project Documentation (Page 891)
How to Open Page View (Page 891)
How to Set the Print Parameters (Page 890)

2.11.11.2 How to Set the Print Parameters

Introduction

You can modify the printout as follows:

- By specifying a layout that differs from the standard layout
- By selecting a page range
- By selecting a printer selection
- By printing to file

Requirement

At least one edit window must be open.

Procedure

1. In the Global Script menu bar, open the "File" menu.
2. Select "Project Documentation Setup..."
3. In the subsequent dialog, adjust the desired settings.
4. Apply the settings by clicking "OK".

---

**Note**
The settings are automatically saved and are not reset when WinCC is restarted.

---

### 2.11.11.3 How to Open Page View

**Introduction**
Before you begin printing a function or action, it is sometimes an advantage to first see a preview of the printout on the screen (in page view).

The content of the active edit window is displayed in page view.

**Procedure**
1. In the Global Script menu bar, open the "File" menu.
2. Select "Project Documentation Setup...".

---

### 2.11.11.4 How to Print the Project Documentation

**Introduction**
You can output the contents of the active edit window to a printer or to a file. The selected print parameter settings are applied.

**Procedure**
1. In the Global Script menu bar, open the "File" menu.
2. Select "Print Project Documentation".
2.12 Creating and Editing Functions

2.12.1 Creating and Editing Functions

Introduction

The system distinguishes between project, standard and internal functions. WinCC is delivered with a broad selection of standard and internal functions. Furthermore, you can create your own project and standard functions or modify standard functions. Please remember, however, that the standard functions included with WinCC are overwritten when WinCC is reinstalled so any modifications are lost.

Internal functions cannot be created or edited.

Using Functions

If the same calculation must be performed - with different starting values - in several actions, it would be to your advantage to program a function to perform this calculation. Subsequently you can simply call this function with the current parameters in the actions.

This approach has a number of advantages:

• The code is only programmed once.
• Modifications are only made at one point, namely in the procedure, not in each action.
• The action code is shorter and, thus clearer.
Finding Functions

To access existing functions or create new ones, you can either use the "File" menu in the Global Script navigation window or click the corresponding button in the toolbar.

Functions are stored in the file system as follows:

```
<table>
<thead>
<tr>
<th>Project path</th>
</tr>
</thead>
<tbody>
<tr>
<td>library</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Installation path</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiLib</td>
</tr>
<tr>
<td>Alarm</td>
</tr>
<tr>
<td>Graphics</td>
</tr>
<tr>
<td>Report</td>
</tr>
<tr>
<td>TagLog</td>
</tr>
<tr>
<td>WinCC</td>
</tr>
<tr>
<td>Windows</td>
</tr>
</tbody>
</table>
```
Editing and Compiling Functions

A function is edited and compiled in its own edit window. The edit window displays messages from the compiler after a compilation run. These might be warnings or error messages. In each case, a summary of the number of warnings and error messages is output.

What Happens When Functions Are Renamed?

In the navigation window, functions are always displayed with their function not their Windows file names. If you change the name of a function and then compile it, the name displayed in the navigation window longer match the function name. This is indicated in the navigation window with the prefix “*” (asterisk) in front of the name. As soon as you save the function, the current function name is displayed in the navigation window.

Note re. Saving Functions

If you save a function that has not been properly compiled, the symbol is shown in the navigation window.

If you save a function that has not been compiled without errors, the symbol is shown in the navigation window.

See also

Working in the Edit Window (Page 880)
How to Use Functions from Other Sources (Page 903)
How to Rename a Function (Page 902)
How to Compile and Save a Function (Page 901)
How to Protect a Function Against Unauthorized Access (Page 900)
Inserting Additional Function-Related Information (Page 898)
How to Use Standard and Project Functions (Page 897)
How to Use Internal Functions (Page 897)
How to Write Function Code (Page 896)
How to Create a New Function (Page 895)
Characteristics of Standard Functions (Page 868)
Characteristics of Project Functions (Page 867)
2.12.2 How to Create a New Function

Introduction

The procedure is identical for both project and standard functions. In the navigation window, specify the type (project or standard function) and, for standard functions, the group, e.g. "graphics". This also specifies the place where the file is to be saved.

Global Script suggests a default name, e.g. "new_function_3", for the new function. This is also the file name of the function. To ensure that the function name is unique, the suggested name includes a sequential number.

As a rule, the default name should be replaced with a more informative function name. When the renamed function is first saved, the file name can also be changed.

Global Script adds the following information to the function: date created, date modified and version. This information can be viewed in the "Properties" dialog. In the same dialog, you can also assign a password to protect the function against unauthorized modification or viewing. To open the dialog, click the button.

Note

The characters supported by ANSI-C are also supported for the function name:

- Letters, apart from regional special characters
- Numbers
- Underscore

Procedure

1. In the navigation window, open the shortcut menu for the desired group.
2. Select "New"
   - If a new function has been created, the first line of code in the associated edit window contains the type of return value and the default name of the new function. In the brackets following this, you can enter transfer parameters if you wish.
   - The function code is entered between the curly brackets.

Alternative operation

Alternatively, you can also create a new function by clicking the associated button in the toolbar, via the "File" menu or by using the corresponding key combination.

See also

Inserting Additional Function-Related Information (Page 898)
How to Use "Save As..." (Page 886)
2.12.3 How to Write Function Code

Introduction

The function code is written in the edit window for the function. The programming language is ANSI-C.

The code of any project or standard function can call other functions. The called function can be a project, standard, internal or DLL function. To make certain that the called function is known to the calling function, the line #include "apdefap.h" is added as the first line of code in the calling function code.

In the navigation window under "Internal Functions", the C-function library is available as "c_bib".

The first line of code contains the type of the returned value and the default name of the new function. Parameters can be passed by entering them in the following brackets.

The function code is entered between the braces.

Procedure

1. Double-click the function in the navigation window to open it in an edit window.
2. Set the cursor where you wish to begin writing.
3. Enter the desired code.

Alternative operation

You can also open a function as follows:

In the navigation window, open the shortcut menu for the desired action and click "Open" or "File\Open...". You can also click the button in the standard toolbar or use the corresponding key combination.

Note

A maximum of 32 Kbytes of memory is available for local tags (tags defined within the braces of the function code).

See also

How to Use Standard and Project Functions (Page 897)
How to Use Internal Functions (Page 897)
Editing Functions with the Mouse (Page 882)
Editing Functions with the Keyboard (Page 881)
Working in the Edit Window (Page 880)
2.12.4 How to Use Internal Functions

Introduction

You can use any of the internal functions as part of your function code. The internal functions are found in the navigation window in the "Internal Functions" group.

If you have used the "Assigning Parameters" dialog to add a function, the function's comments show the type of value returned.

Procedure

1. Place the cursor at the point at which the internal function is to be inserted.
2. In the navigation window, open the shortcut menu for the internal function to be added.
3. Select "Assigning Parameters". The "Assigning Parameters" dialog is opened. This dialog has one line for each parameter. In the "Value" column, enter the respective current parameter.
4. In the "Value" column, enter the current value for each of the required parameters. This can be accomplished by either a direct entry from the keyboard, or you can open the menu in the "Value" column (single-click and then click the displayed button). From the menu, you can open the selection dialog for tags, pictures or graphic objects.
5. Confirm your entries with "OK". The parameterized function is inserted in the edit window at the location of the cursor.

Alternative operation

Alternatively you can also open the "Assigning Parameters" dialog for an internal function by double-clicking the function to be added.

Note

If you close the "Assigning Parameters" dialog with "OK" without entering the current parameter value, the internal function is inserted with its formal parameters. You can then set the parameters in the edit window at a later stage.

Instead of using the "Assigning Parameters" dialog, you can also use the keyboard to enter the function.

2.12.5 How to Use Standard and Project Functions

Introduction

You can use any project or standard function as part of the function code, if you have first added the line #include "apdefap.h" in the header. The project functions are found in the navigation window in the "Project Functions" group. The standard functions are found in the navigation window in the "Standard Functions" group.
Project functions are entered in the "Ap_pbib.h" header file; standard functions are entered in the "Ap_glob.h" header file. These entries are made by the system. The "Ap_glob.h" header file is integrated into the "Ap_pbib.h" header file. The "Ap_pbib.h" header file itself is linked to the "Apdefap.h" header file. Therefore all project and standard functions are declared in the "Apdefap.h" file header.

To inform the compiler of the project and standard functions added, add the line `#include "apdefap.h"` as the first line in the function code.

If you have used the "Assigning Parameters" dialog to add a function, the function comments show the type of value returned.

### Procedure

1. Place the cursor at the point at which the project or standard function is to be inserted.
2. In the navigation window, open the shortcut menu for the function to be added.
3. Select "Assigning Parameters". The "Assigning Parameters" dialog is opened. This dialog has one line for each parameter. In the "Value" column, enter the respective current parameter.
4. In the "Value" column, enter the current value for each of the required parameters. This can be accomplished by either a direct entry from the keyboard, or you can open the menu in the "Value" column (single-click and then click the displayed button). From the menu, you can open the selection dialog for tags, pictures or graphic objects.
5. Confirm the entries by clicking "OK".

---

**Note**

If the function does not require a parameter, it is added to the function code immediately without opening the "Assigning Parameters" dialog.

If you close the "Assigning Parameters" dialog with "OK" without entering the current parameter value, the internal function is inserted with its formal parameters. You can then set the parameters in the edit window at a later stage.

---

### 2.12.6 Inserting Additional Function-Related Information

**Introduction**

Additional information can be assigned to every function.

When a new action is created, the creation date is automatically entered in the function-related information and is unchangeable. In addition, the version number 1.0 is also assigned to the number. The version numbers can be individually assigned when editing a function. When a function is changed and saved, the current date of change is entered automatically and is unchangeable. This dialog can be used to assign a password to protect the function from unauthorized viewing and modification.
Requirements

The function to which the information relates must be open in an edit window.

Procedure

1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the required entries.
3. Confirm the entries by clicking "OK".

Alternative operation

Alternatively, the "Properties" dialog can be opened as follows:

Click the "Edit" menu and select the "Info" option or use the corresponding key combination.

See also

How to Protect a Function Against Unauthorized Access (Page 900)
How to Set Different Views (Page 885)
Working with the Toolbars (Page 882)
2.12.7 How to Protect a Function Against Unauthorized Access

Introduction

Functions can be protected with a password against unauthorized read and write access. The password is a part of the function-related information.

Requirements

The function to be compiled must be opened in the edit window.

Procedure

1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Password" check box.
3. Click the "Change" button.
4. Enter the password in the "Password" field.
5. Enter the password again in the "Confirmation" field.
6. Confirm your entries with "OK".
7. Click "OK" to close the dialog.

Alternative operation

Alternatively, the "Properties" dialog can be opened as follows:
Click the "Edit" menu and select the "Info" option or use the corresponding key combination.

Note

A password-protected function can only be opened in the edit window if the correct password is entered.
To deactivate the password protection, clear the "Password" check box.
See also

How to Set Different Views (Page 885)
Working with the Toolbars (Page 882)

2.12.8 How to Compile and Save a Function

Introduction

In order to use a function, it must first be compiled. Only the function in the active edit window is compiled.

Errors reported by the compiler are displayed in the lower portion of the window. Each message is displayed on a separate line. The line includes the line number in the source code where the error occurred, a hexadecimal encoded error code and a description of the error.

Double-click such a line to select the source code line where the error occurred.

It is recommended that you examine the first message error listed, as subsequent ones could be errors resulting from the first one. If the first error is corrected, then the others might disappear after the next compilation.

To make the changes permanent, the function must be saved.

If you save a function that has been compiled with errors, or not at all, the icon is assigned to this function in the navigation window.

If you save a function that has been compiled without errors, the icon is assigned to this function in the navigation window.

Requirements

The function to be compiled must be opened in the edit window.

Procedure

1. Use the toolbar to set the language for compilation of the C function.
2. Click on the "Edit" toolbar.
3. Examine the compiler messages in the lower portion of the edit window.
4. If the compiler reported an error, the function code must be corrected. After this has been done, start again with step 1 in this table.
5. If the compiler generated warnings, the function source code may require correction. After the code has been corrected, start again with step 1 in this table, otherwise proceed to step 6.
6. Click on the "Default" toolbar.
Alternative operation

Alternatively, the compilation process can be initiated in the following ways:
Select "Compile" from the "Edit" menu, select the "Compile" option from the shortcut menu of the edit window or use the corresponding key combination.

Saving may also be performed in the following ways:
Select "Save" from the "File" menu or use the corresponding key combination.

Note

The compiler does not output an error message if tag names are used several times in a C function. This is also the case if a tag name is used both as transfer parameter and as local tag definition.

For example, the following faulty script does not trigger an error message in the compiler:

```c
void neue_Funktion(DWORD dwMyVar)
{
    DWORD dwMyVar = 0;
}
```

Message in the output window of the compiler:

Compiling ...

0 Error(s), 0 Warning(s)

See also

Runtime Behavior of Actions (Page 927)

2.12.9 How to Rename a Function

Introduction

It is recommended that you rename the function, when it is created.

The name of the function in the edit window is then changed accordingly. Since this also changes the code, the function must be recompiled. The old function name, displayed in the navigation window, is assigned prefix "*" (asterisk).
Afterwards, the modified function must be saved at which time you can change the path and file name. The old function should then be deleted to avoid accumulating a collection of obsolete functions.

**Note**

Please note that only certain characters may be used in function names: characters (with the exception of national special characters), numbers and the underscore.

**Procedure**

1. Change the function name in the edit window.
2. Click the button in the editing toolbar. The function is compiled.
3. Click the button in the standard toolbar to save the function.
4. If desired, enter a different path and/or file name.
5. Confirm your entry by clicking "Save".

**Alternative operation**

Alternatively, the compilation process can be initiated in the following ways:

Select "Compile" from the "Edit" menu, select the "Compile" option from the shortcut menu of the edit window or use the corresponding keyboard shortcut.

Saving may also be performed in the following ways:

Select "Save" from the "File" menu or use the corresponding key combination.

**2.12.10 How to Use Functions from Other Sources**

**Introduction**

Project functions from other WinCC projects and standard functions from other WinCC systems can also be made useable for the current project. To do so they must be brought into the current project.

Apart from the location in the file system in which they are saved, there is no difference in the procedure for project or standard functions.

**Procedure**

1. Copy the functions. Project functions are copied in the "library" directory for the WinCC project. Standard functions are copied in the "aplib..." directory in the WinCC path. The content of the navigation window is updated automatically.

2. Click in the "Standard" toolbar. When the header is regenerated, the copied functions are registered so that you can use them in your current project.
**Alternative operation**

Alternatively, you can start a generation process as follows:

Open the "Options" menu and select "Regenerate Header" or use the corresponding key combination.

---

**Note**

In the event that WinCC is reinstalled or upgraded, the standard functions that were modified are deleted or replaced by the unedited standard functions.

If WinCC is in Runtime, the Runtime system is not influenced by the regeneration of the header.
2.13 Creating and Editing Actions

2.13.1 How To Create and Edit Actions

Introduction

There system distinguishes between global and local actions. In a client-server project, global actions are carried out on all computers in the project, whereas local ones are carried out only on the computer to which they are assigned.

A global action can, for instance, be used to perform a calculation on all computers in the project.

An example of use for a local action might be to output a log file on a server.

The process of creating and editing both action types is identical.

Differences between actions and functions

- Actions, in contrast to functions, can have a trigger. This means that a function, by itself, cannot be executed in Runtime.
- Actions can be exported and imported.
- Authorizations can be assigned to actions. This authorization refers to the operational options for the Global Script Runtime troubleshooting window.
- An action has no parameters.

Finding actions

New actions can be created, and existing actions can be accessed through the Global Script navigation window.

Actions are stored in the file system as follows:
Editing and Compiling Actions

An action is edited and compiled in a separate edit window. The edit window displays messages from the compiler after a compilation run. These might be warnings or error messages. In each case, a summary of the number of warnings and error messages is output.

Display of Actions

If a syntactically incorrect action is stored, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with no trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with a trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

Please note the following when creating actions:

The CrossReference feature of WinCC allows for the creation of cross-references. When building the cross-reference list, in order that the tags and images be recognized when function calls are used as part of actions, the coding rules described further below are to be observed.

Renaming actions

Actions are always displayed with their file names in the navigation window. Renaming an action means renaming the file containing the action code.

System behavior if actions are changed, deleted and saved at Runtime

If a local action is stored at runtime, then all local and global actions of the computer are reset on the computer to which the local action belongs.

If a global action is stored in Runtime, then all local and global actions for the entire project – and thus on all computers – are reset.
Such a reset might reinitialize for examples tags and times that are used as triggers for actions, triggering the action at that stage.

Static tags used in the reset actions are reinitialized.

Possible causes for an action not being performed in Runtime

Failure of an action to be executed in Runtime might have the following causes:

- The action has no trigger
- The action was not compiled
- Global Script Runtime is not enabled in the project start list

Note

Before creating an action, check whether the relevant functionality can also be implemented on the automation device.

See also

- How to Protect an Action Against Unauthorized Access (Page 912)
- How to Use Actions From Other Sources (Page 925)
- How to Rename an Action (Page 924)
- How to Import an Action (Page 923)
- How to Export an Action (Page 922)
- How to Assign Authorizations (Page 922)
- Triggers (Page 914)
- How to Compile and Save an Action (Page 913)
- How to add action-related information (Page 910)
- How to Edit Actions (Page 910)
- How to Create a New Action (Page 909)
- WinCC Coding Rule (Page 908)
- How to Add Global Script Runtime to a Project's Startup List (Page 873)
- Characteristics of Global Actions (Page 872)
- Characteristics of Local Actions (Page 871)
2.13.2 WinCC Coding Rule

Coding Rules for the Use of CrossReference

The CrossReference feature of WinCC allows for the creation of cross-references. To ensure that the software can recognize the tags and pictures used in function calls made within actions, the coding rules given here must be observed.

The action's code begins with two sections. In the first section, you must declare all tags used; in the second section all picture names used.

Do not enter any other instructions in the sections.

Both sections are already present in the form of comments when an action is created:

```
// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END
```

The sections are expanded, for example as follows:

```
// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
#define ApcTagName1 "TagName1"
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
#define ApcPicName1 "PicName1"
#define ApcPicName2 "PicName2"
#define ApcPicName3 "PicName3"
// WINCC:PICNAME_SECTION_END
```
Calls to functions to read and write tags and the utilization of picture names must then be handled using the defined names:

```c
GetTagDWord(ApcTagName1);
OpenPicture(ApcPicName1);
SetPictureName(ApcPicName2,"PictureWindow1",ApcPicName3);
```

2.13.3 How to Create a New Action

Introduction

In a client-server project, global actions are carried out on all computers in the project, whereas local ones are carried out only on the computer to which they are assigned.

The procedure is identical for both global and local actions. By specifying, in the navigation window, the location in which the action is saved, you specify its type (global or local).

Global Script suggests a default name for the new action.

A newly created action already contains the instruction `#include "apdefap.h"`. Therefore, all functions are registered within the action. The name of the action is found in the third line. The first three lines cannot be deleted nor modified. This means that every function can be called from each action without requiring any special measures. Furthermore every action has a returned value of type "int" and it is already set to a value of 0.

A returned value of an action can be used in conjunction with GSC Runtime for diagnostic purposes.

The action code begins with a code framework in the form of comments. If this coding framework is filled out in accordance with the coding rules, the tags and picture names are recognized by CrossReference.

Procedure

1. In the navigation window, open the shortcut menu for the desired action type.
2. Select "New".

Alternative operation

Alternatively, you can also create a new action by clicking the associated button in the toolbar, via the "File" menu or by using the corresponding key combination.

See also

GSC Runtime (Page 928)
WinCC Coding Rule (Page 908)
2.13.4 How to Edit Actions

Introduction

An action is edited in its own edit window exactly like a function. Only the first three lines cannot be edited.

The action must have a return value. The returned value is of the type "int" and is preset to 0. A returned value of an action can be modified and used in conjunction with GSC Runtime for diagnostic purposes. The returned value's type cannot be changed.

To execute an action in Runtime, the action must have a trigger.

Procedure

1. Double-click the action in the navigation window to open it in an edit window.
2. Edit the action code.

Alternative operation

You can also open an action as follows:

In the navigation window, open the shortcut menu for the desired action and click "Open" or "File\Open...". You can also click the button in the standard toolbar or use the corresponding key combination.

Note

A maximum of 32 Kbytes of memory is available for local tags (tags defined within the braces of the action code).

See also

GSC Runtime (Page 928)
How to Write Function Code (Page 896)

2.13.5 How to add action-related information

Introduction

Additional information can be assigned to every action.

When a new action is created, the creation date is entered in the action-related information automatically and is unchangeable. The action is also assigned version number 1.0. The version numbers can be individually assigned when editing an action. When an action is changed and saved, the current date of change is entered automatically and is unchangeable.
This dialog can be used to assign a password to protect the action against unauthorized read and write access.

Requirements

The action to which the information relates must be open in an edit window.

Procedure

1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the required entries.
3. Confirm your entries with "OK".

Alternative operation

Alternatively, the "Properties" dialog can be opened as follows:
Click the "Edit" menu and select the "Info" option or use the corresponding key combination.

See also

How to Protect an Action Against Unauthorized Access (Page 912)
How to Set Different Views (Page 885)
Working with the Toolbars (Page 882)
2.13.6 How to Protect an Action Against Unauthorized Access

Introduction
Actions can be protected with a password against unauthorized read and write access. The password is a part of the action-related information.

Requirements
The action to be compiled must be opened in the edit window.

Procedure
1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Password" check box.
3. Click the "Change" button.
4. Enter the password in the "Password" field.
5. Enter the password again in the "Confirmation" field.
6. Confirm your entries with "OK".
7. Click "OK" to close the dialog.

Alternative operation
Alternatively, the "Properties" dialog can be opened as follows:
Click the "Edit" menu and select the "Info" option or use the corresponding key combination.

Note
A password-protected action can only be opened in the edit window if the correct password is entered.
To deactivate the password protection, clear the "Password" check box.
2.13.7 **How to Compile and Save an Action**

**Introduction**

In order to use an action, it must first be compiled. Only the action in the active edit window is compiled.

Errors reported by the compiler are displayed in the lower portion of the window. Each message is displayed on a separate line. The line includes the line number in the source code where the error occurred, a hexadecimal encoded error code and a description of the error.

Double-click such a line to view the source code line where the error occurred.

It is recommended that you examine the first message error listed, because subsequent ones could be errors resulting from the first one. If the first error is corrected, then the others might disappear after the next compilation.

**Requirement**

The action to be compiled must be opened in the edit window.

**Procedure**

1. Set the language for C compilation using the toolbar.
2. Click on the "Edit" toolbar.
3. Examine the compiler messages in the lower portion of the edit window.
4. If the compiler reported an error, the action source code must be corrected. After this has been done, start again with step 1 in this table.
5. If the compiler generated warnings, the action source code may require correction. After the code has been corrected, start again with step 1 in this table, otherwise proceed to step 6.
6. Click on the "Default" toolbar.

**Alternative operation**

Alternatively, the compilation process can be initiated in the following ways:

Select "Compile" from the "Edit" menu, select the "Compile" option from the shortcut menu of the edit window or use the corresponding keyboard shortcut.

Saving may also be performed in the following ways:

Select "Save" from the "File" menu or use the corresponding key combination.
Display of Actions

If a syntactically incorrect action is stored, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with no trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with a trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

See also

Runtime Behavior of Actions (Page 927)

2.13.8 Triggers

2.13.8.1 Triggers

Defining and Using Triggers

Triggers are used to execute actions in Runtime. To do this, a trigger is linked to an action, forming the triggering event for calling the action. Actions without triggers are not executed.

Trigger Types

The following trigger types are available:

Acyclic Triggers

These consist of a specified date and time. The action specified by such a trigger is performed once at the date and time specified.

Cyclic Triggers

These consist of a specified time interval and starting point. The following types of cyclic triggers are available:

- Default cycle. The start of the first time interval coincides with the start of Runtime. The length of the interval is determined by the cycle.
- Hourly. The start of the interval is specified as minutes and seconds. The length of the interval is an hour.
• Daily. The start of the interval is specified by the time (hours, minutes and seconds). The length of the interval is a day.

• Weekly. The start of the interval is specified by the day of the week (Monday, Tuesday, etc.) and the time. The length of the interval is a week.

• Monthly. The start of the interval is specified by the day and time. The length of the interval is a month.

• Annually. The start of the interval is specified by the day, month and time. The length of the interval is a year.

**Tag Triggers**

These consist of the specification of one or more tags. The action associated with such a trigger is performed each time a change in the value of one of these tags is detected. How the tag values are queried may be customized for each tag. Either cyclic polling with a specified period or a reaction as soon as the system detects a change in the tag value may be selected. Depending on the choice of query method, it is possible that the tag changes but the system does not detect this. In this case the action is not performed.

**Effect of Triggers on an Action**

If the action is associated with only one trigger, then the action is performed as soon as the triggering event occurs.

However, an action may be associated with multiple triggers, such as a cyclic trigger and a tag trigger. In this case, the action is performed whenever one of the two triggering events occurs. If two events occur simultaneously, the action is executed twice in sequence. If two tag triggers fire at the same time, the action is performed only once.

The processing of an action should be completed before another call of the action occurs, as there might otherwise be an overflow of the queue.

**Tip:** If the action is not to be carried out with each event occurrence, then a condition can be specified for the action that controls its further course dependent on the event. If the action is not to be executed any more, it can be terminated with a <value> return.

**Rules for the Selection of Triggers**

Depending on the system, it cannot be guaranteed that an action with a cyclic trigger is carried out at exactly the specified time. If this is a requirement, then the task (such as a check, etc.) must be implemented on the automation device.

For processing, tag triggers have priority over cyclic triggers.

• For a cyclic trigger, the action is always performed when the trigger event occurs, e.g. every 20 seconds. (✔ = action is performed)
The tag trigger only causes the action to be performed if the value of the trigger tag has changed. This is done to reduce the system load ( действие выполняется). The values of tags contained in the trigger are already known when the action begins. The GetTag() call can be used to access the value directly. Processing is much faster than for trigger tags than for those not contained in the trigger whose values must be obtained via GetTag() requests.

Display of Actions

If a syntactically incorrect action is stored, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with no trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

If a syntactically correct action is stored with a trigger, it is displayed in the Global Script navigation window with the icon shown to the left.

See also

How to delete a trigger (Page 921)
How to change a trigger (Page 920)
How to Add a New Trigger of the "Tag" Type (Page 918)
How to Add a New Trigger of the "Timer" Type (Page 917)
2.13.8.2 How to Add a New Trigger of the "Timer" Type

Introduction
Triggers are used to execute actions in Runtime. To do this, a trigger is linked to an action, forming the triggering event for calling the action. Actions without triggers are not executed.

"Timer" type triggers can be cyclic or acyclic triggers.

Acyclic triggers consist of a specified date and time. The action specified by such a trigger is performed once at the date and time specified.

Cyclic triggers consist of a specified time interval and starting point. The following types of cyclic triggers are available:

- Default cycle. The start of the first time interval coincides with the start of the Runtime system. The length of the interval is determined by the cycle.
- Hourly. The start of the interval is specified as minutes and seconds. The length of the interval is an hour.
- Daily. The start of the interval is specified by the time (hours, minutes and seconds). The length of the interval is a day.
- Weekly. The start of the interval is specified by the day of the week (Monday, Tuesday, etc.) and the time. The length of the interval is a week.
- Monthly. The start of the interval is specified by the day and time. The length of the interval is a month.
- Annually. The start of the interval is specified by the day, month and time. The length of the interval is a year.

Requirement
The action to be linked with a trigger must be open in the active edit window.

Procedure
1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Trigger" tab.
3. Select the trigger source "Timer" and click the "Add" button.
4. Select "Single" to add an acyclic trigger or select a cycle to add a cyclical trigger.
5. Complete the required details in the dialog.
6. Confirm the entries by clicking "OK".
7. Close the "Properties" dialog by clicking "OK".

Alternative operation
Alternatively, the "Properties" dialog can be opened as follows:
In the "Edit" menu, select "Info", select "Info / Trigger" in the shortcut menu of the edit window, or use the corresponding key combination.

2.13.8.3 How to Add a New Trigger of the "Tag" Type

Introduction
Triggers are used to execute actions in Runtime. To do this, a trigger is linked to an action, forming the triggering event for calling the action. Actions without triggers are not executed.

Tag triggers consist of one or more specified tags. The action associated with such a trigger is performed each time a change in the value of one of these tags is detected. How the tag values are queried may be customized for each tag. Either cyclic polling with a specified period or a reaction as soon as the system detects a change in the tag value may be selected. Depending on the choice of query method, it is possible that the tag changes while the system fails to detect this. In this case the action is not performed.

Requirements
The action to be linked with a trigger must be open in the active edit window.

Procedure
1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Trigger" tab.
3. Select the trigger source "Tag" and click the "Add" button. The "Add Tags" dialog is opened.

4. Click the button to open the tag selection dialog, select a tag and confirm your selection by clicking "OK".

5. In the "Add Trigger" dialog, open the shortcut menu in the "Standard cycle" column and then select the desired monitoring cycle. Selecting "After Every Change" results continuous monitoring.

6. Repeat steps 4 and 5, if you want add more tags.

7. Confirm the entries by clicking "OK".

8. Close the "Properties" dialog by clicking "OK".

Alternative operation

Alternatively, the "Properties" dialog can be opened as follows:

In the "Edit" menu, select "Info", select "Info / Trigger" in the shortcut menu of the edit window, or use the corresponding key combination.

In the "Add Trigger" dialog, you can also enter a tag name directly and insert the tag in the "Trigger Name" column by clicking "Add". With this approach, the system does however not check whether the tag exists.
2.13.8.4 How to change a trigger

Introduction
Triggers that have been defined can be changed at any time. They can also be changed in Runtime.

Requirements
The relevant action must be opened in the edit window.

Procedure
1. Click the \[button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Trigger" tab and select the trigger you wish to change.
3. Click the "Change" button to open the "Change Trigger" dialog.
4. Make the desired changes.
5. Confirm the entries by clicking "OK".
6. Close the "Properties" dialog by clicking "OK".

Alternative operation
Alternatively, the "Properties" dialog can be opened as follows:
In the "Edit" menu, select "Info", select "Info / Trigger" in the shortcut menu of the edit window, or use the corresponding key combination.
2.13.8.5 How to delete a trigger

Introduction

Triggers that have been defined can be deleted at any time. They can also be deleted in Runtime.

If a trigger is deleted in Runtime, it only takes effect after the action is saved.

Requirements

The relevant action must be opened in the edit window.

Procedure

1. Click the button in the editing toolbar. The "Properties" dialog is opened.
2. Select the "Trigger" tab and select the trigger you wish to delete.
3. Delete the selected trigger by clicking "Delete".
4. Close the "Properties" dialog by clicking "OK".

Alternative operation

Alternatively, the "Properties" dialog can be opened as follows:

In the "Edit" menu, select "Info", select "Info / Trigger" in the shortcut menu of the edit window, or use the corresponding key combination.
2.13.9 How to Assign Authorizations

Introduction

With the Global Script - Runtime diagnostic tool, you can influence the processing of actions in Runtime. Each action can be assigned an authorization. This authorization only effects the operation in the Global Script - Runtime window.

Requirement

The relevant action must be opened in the editing window.

Procedure

1. Click in the toolbar.
   The "Authorizations" dialog box is opened.
2. Select an authorization.
3. Confirm your selection with "OK".

See also

GSC Runtime (Page 928)

2.13.10 How to Export an Action

Introduction

Use export and import to move actions between projects. The triggers linked to the actions are retained in the process.
Requirements

The action to be exported must be opened in the edit window.

Procedure

1. Click the button in the editing toolbar. The "Save As" dialog is opened.
2. Select the path and file name for the action that you wish to export.

3. Close the dialog by clicking the "Save" button.

Alternative operation

Alternatively, you can start the export as follows:

In the "Edit" menu, select "Export", select "Export" in the shortcut menu of the edit window, or use the corresponding key combination.

2.13.11 How to Import an Action

Introduction

Use export and import to move actions between projects. The triggers linked to the actions are retained in the process.
The action in the active edit window is replaced by the imported action.

**Procedure**

1. Click the button in the editing toolbar. The "Open" dialog is opened.
2. Select the path and file name of the action that you wish to import.
3. Close the dialog by clicking the "Open" button.

**Alternative operation**

Alternatively, you can start the import as follows:

In the "Edit" menu, select "Import", select "Import" in the shortcut menu of the edit window, or use the corresponding key combination.

**2.13.12 How to Rename an Action**

**Introduction**

You may rename an action at any time. The action is thereby assigned a different file name.
Requirements

The action to be renamed may not be open in the edit window.

Procedure

1. Open the shortcut menu of the action to be renamed.
2. Select "Rename".
3. Enter a new name with the file extension ".pas".
4. Confirm the new name by pressing the <ENTER> key.

Alternative operation

Alternatively, click the action name twice to complete the renaming.

Note

If you do not enter the file extension ".pas", the action name remains unchanged.

2.13.13 How to Use Actions From Other Sources

Introduction

You have two options for using actions from other sources in your project:

- Importing exported actions
- Copy the file with the desired action into the corresponding path in your project. The path for local actions is "<Computer_name>\Pas" in the project path. The path for global actions is "\Pas" in the project path.

To show the copied actions in the navigation window, you must refresh the display. This can be accomplished by quitting and then restarting Global Script.
If Runtime is active, imported actions are only executed, after they have been opened in Global Script Editor and then saved.

**Note**

Actions can include calls to project and standard functions. These can in turn have calls to project and standard functions etc. Therefore, when importing actions from other sources, you must make sure that the current project has all necessary functions.

Particular attention is called for whenever the actions were imported from a different computer. Since standard functions can be customized by the user, it is possible that the standard functions called in the action have a different functionality on the source computer than the ones with the same names on the target computer.

**See also**

How to Import an Action (Page 923)
How to Export an Action (Page 922)
2.14 Runtime Behavior of Actions

2.14.1 Runtime Behavior of Actions

Analysis of Runtime Behavior

WinCC provides a range of tools with which the Runtime behavior of actions can be analyzed. These are the application windows GSC Runtime and GSC Diagnose plus the application apdiag.exe.

To use the application windows GSC Runtime and GSC Diagnose, they must be added to a process picture. This may be a process picture made especially for diagnostic purposes. This picture is then called in Runtime.

The application windows are used for the following strategies:

- GSC Runtime supplies information about the dynamic behavior of all (Global Script) actions, supports the individual start and the Start and End Action for each individual action and provides an entry point into Global Script Editor, while Runtime is active.

- GSC Diagnose outputs the printf instructions (contained in the actions) in the order in which they are called. This also applies to the printf instructions in functions that are called in actions. Through a well thought out use of printf instructions, for example to output the values of tags, it is possible to follow the action's flow and that of the called functions. Even error conditions, which result in a call to the OnErrorExecute function, are displayed in the GSC Diagnose window.

Note

When using dynamic C-scripts with access to picture objects, you should note that execution of the script is not terminated automatically by closing the picture.

This scenario may to the failure of access to an object that is addressed in the script, e.g. if properties of the type "Text" are read and the values returned are modified or processed in subsequent string operations.

See also

GSC Diagnose (Page 933)

GSC Runtime (Page 928)
2.14.2 GSC Runtime

2.14.2.1 GSC Runtime

GSC Runtime Window

GSC Runtime is a window showing the dynamic behavior of all (Global Script) actions in Runtime. Additionally, GSC Runtime, enables you to influence the execution of each individual action and provides an entry point into Global Script Editor, while Runtime is active.

The following information is output:

- **Action**: Name of the action
- **ID**: Action ID. It is used internally to the system and for example is output by the function `OnErrorExecute` together with a description of the error in the event that an error occurs in the action. GSC Runtime supplies the name of the action with this ID. The connection between the ID and action name ceases to be valid when Runtime is terminated or an action is saved while Runtime is active.
- **Status**: Current status of the action. For possible statuses, refer to the table below.
- **Activation Interval**: The time in the form hours:minutes:seconds that has elapsed between two calls to the action.
- **Return Value**: Return value of the action.
- **Started On**: Date and time at which the current action was started.
- **Next Start**: Date and time at which the action is started again.
- **Error Message**: Contains the error text in the case of an error.

### Statuses of actions

Possible statuses of actions:

- Action was started.
- Action was ended.
- Action was stopped.
- Action is running.
Error during start of action!
Error during execution of action!

Error messages

Possible error messages:

- No error occurred.
- The application is already connected to the script control. No additional connection setup is possible.
- There is no connection to the script control. Possibly no connection setup took place.
- An error occurred during interprocess communication. The cause of the error is unknown.
- Undefined error.
- The parameter assignment is wrong. Some necessary parameters may be missing.
- Script control is not started. Check has whether WinCC has been started.
- Time-out occurred. Check the connection or increase the monitoring time.
- Script control was terminated.
- The service channel could not be installed.
- An unknown job number was used for the EndAct job.
- The action could not be executed without error. The returned results are invalid.
- An error occurred in the server application.
- The maximum number of connections to the script control has been reached.
- The transaction is unknown. An attempt was made to terminate a transaction that was not logged on previously.
- A pre-compiled header file cannot be generated from a pre-compiled header file.
- There is no access to the action. The module is being used presently.
- The program is invalid.
- The action is invalid.
- The script control could not set up the file.
- The script interpreter does not have enough memory.
- The file format is invalid for the script control.
- The script control could not open the file.
- The program is presently locked by the script control. No further access is possible.
- The action has already been given to the script control for processing.
- In this action, a conflict has occurred with another action.
- The script control could not find the action.
- The script control could not find the function.
- The specified line information is invalid.
The specified symbol is outside the valid range.

The provided memory is too small for the script interpreter.

The script interpreter does not recognize the specified type.

The specified symbol was not found.

Load project functions.

A stack overflow has occurred in the script interpreter during execution. Further execution of the action is being canceled.

Division by 0 occurred during execution of an action. The action is being canceled.

Within the action, a reference was made during execution to a symbol that does not exist.

Within the action, an attempt was made during execution to access an undefined memory area.

The script interpreter ran into a breakpoint.

The script interpreter was advanced in the debugger by one processing step.

The action contains no interpreter code.

The action has the wrong data format.

The return value of the action cannot be represented as a variant.

There is insufficient memory to execute this operation.

An error has occurred within the transaction. For more information, see the AP_ACT_KEYs.

An error occurred while executing the action. For more information, see the AP_ACT_KEYs.

An error occurred while executing the action. For more information, see the AP_ACT_KEYs.

There is no update capability for the existing data format. The action cannot be read.

**Shortcut Menu for Actions**

The following functions are available for every action in the shortcut menu:

- **End Action**: The corresponding action is not executed again after the current execution is completed.
- **Start Action**: The corresponding action is executed again when the next trigger occurs.
- **Start**: The relevant action is executed once.
- **Edit**: The relevant action is opened in the Global Script editor for editing. Runtime remains active. If the edited action is compiled (if necessary) and saved, the changes are immediately adopted by the Runtime system.
You can determine for each action individually, whether or not the popup menu can be opened without a password.

To use GSC Runtime, you must first add an application window of the GSC Runtime type in a process picture. Using the GSC Runtime attributes, you can determine the appearance of the GSC Runtime window.

**Note**

Updating the GSC Runtime window increases the load on the system resources. The system load depends on how many actions are visible in the window. The system load can be lowered by reducing the height of the window so that fewer lines are visible.

**See also**

- How to Edit Actions (Page 932)
- Attributes of GSC Runtime (Page 932)
- How to Place GSC Runtime in a Process Picture (Page 931)
- How to Assign Authorizations (Page 922)

### 2.14.2.2 How to Place GSC Runtime in a Process Picture

**Introduction**

To use GSC Runtime, you must add GSC Runtime to a process picture. This process picture can be an existing picture or a picture that just serves diagnostic purposes. GSC Runtime cannot be added to the process picture directly, rather it must be added as an application in an application window. The application window is itself part of the process picture. The measures described must be performed in Graphics Designer.

**Requirement**

Graphics Designer has been started and the process picture is open.

**Procedure**

1. In the Object palette, select "Smart Object\Application Window".
2. In the drawing area, open the application window.
3. In the "Window Contents" dialog select "Global Script".
4. Confirm the entries by clicking "OK".
5. In the "Template" dialog, select "GSC Runtime".
6. Click "OK" to confirm your selection.
2.14.2.3 Attributes of GSC Runtime

GSC Runtime Window Layout

GSC Runtime has attributes with which you can determine the appearance of the GSC Runtime window in Runtime. These include the geometry attribute and in particular the following attributes:

- **Display**: With this attribute, you can specify whether the window should be visible or hidden. The attribute can be made dynamic with the name "Visible".
- **Sizeable**: Use this attribute to specify whether the window size can be changed in Runtime.
- **Moveable**: Use this attribute to specify whether the window can be moved in Runtime.
- **Border**: Use this attribute to specify whether the window has a border. If the window has a border, its height and width can be changed in Runtime.
- **Title**: Use this attribute to specify whether the window has a title bar.
- **Can be Maximized**: Use this attribute to specify whether the window's title bar has a button to maximize the window in Runtime.
- **Can be Closed**: Use this attribute to specify whether the window's title bar has a button to close the window in Runtime.
- **Foreground**: Use this attribute to specify whether the window is always in the foreground.

The attributes are displayed and can be set in Graphics Designer.

2.14.2.4 How to Edit Actions

Introduction

Each of the actions in your project will be displayed on its own line in the GSC Runtime window. You can open an action in the GSC Runtime window and then edit it using Global Script Editor. After the edited action is saved, it is used in Runtime.

Procedure

1. Open the shortcut menu for the desired action.
2. Select "Edit".

See also

How To Create and Edit Actions (Page 905)
2.14.3 GSC Diagnose

2.14.3.1 GSC Diagnose

Description of Functions

GSC Diagnose outputs the printf instructions (contained in the actions) in the order in which they are called in the Diagnose window. This also applies to the printf instructions in functions that are called in actions. Through a well thought out use of printf instructions, for example to output the values of tags, it is possible to follow the action's flow and that of the called functions. Even error conditions, which result in a call to the OnErrorExecute function, are displayed in the GSC Diagnose window.

![Diagnose window of GSC Diagnose](image)

To use GSC Diagnose, you must first add an application window of the GSC Diagnose type in a process picture. Using the GSC Diagnose attributes, you can determine the appearance of the GSC Diagnose window.

When the picture is changed, the contents in the GSC Diagnose window is deleted.

Note

A printf() may contain maximum 360 characters.
See also

- The Toolbar of GSC Diagnose (Page 935)
- Attributes of GSC Runtime (Page 932)
- How to Place GSC Diagnose in a Process Picture? (Page 934)

### 2.14.3.2 How to Place GSC Diagnose in a Process Picture?

#### Introduction

To use GSC Diagnose, you must add GSC Diagnose to a process picture. This process picture can be an existing picture or a picture that just serves diagnostic purposes. GSC Diagnose cannot be added as an application to the process picture directly, rather it must be added as an application in an application window. The application window is itself part of the process picture. The measures described must be performed in Graphics Designer.

#### Requirement

Graphics Designer has been started and the process picture is open.

#### Procedure

1. In the Object palette, select "Smart Object\Application Window".
2. In the drawing area, open the application window.
3. In the "Window Contents" dialog select "Global Script".
4. Confirm the entries by clicking "OK".
5. In the "Template" dialog, select "GSC Diagnose".
6. Click "OK" to confirm your selection.

See also

- Attributes of GSC Runtime (Page 932)
- The Toolbar of GSC Diagnose (Page 935)
2.14.3.3 Attributes of GSC Diagnose

GSC Diagnose Attributes

GSC Diagnose has attributes with which you can determine the appearance of the GSC Diagnose window in Runtime. These include the geometry attribute and in particular the following attributes:

- **Display:** With this attribute, you can specify whether the window should be visible or hidden. The attribute can be made dynamic with the name "Visible".
- **Sizeable:** Use this attribute to specify whether the window size can be changed in Runtime.
- **Moveable:** Use this attribute to specify whether the window can be moved in Runtime.
- **Border:** Use this attribute to specify whether the window has a border. If the window has a border, its height and width can be changed in Runtime.
- **Title:** Use this attribute to specify whether the window has a title bar.
- **Can be Maximized:** Use this attribute to specify whether the window’s title bar has a button to maximize the window in Runtime.
- **Can be Closed:** Use this attribute to specify whether the window’s title bar has a button to close the window in Runtime.
- **Foreground:** Use this attribute to specify whether the window is always in the foreground.

The attributes are displayed and can be set in Graphics Designer.

2.14.3.4 The Toolbar of GSC Diagnose

Toolbar Functions

The toolbar of GSC Diagnose includes buttons for the control the output in the Diagnose window as well as for saving, printing and opening the contents of the window.

The toolbar contains buttons with the following functions:

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[X]</td>
<td>Deletes the contents of the window.</td>
</tr>
<tr>
<td>[I]</td>
<td>Stops the updating of the window.</td>
</tr>
<tr>
<td>[R]</td>
<td>Resumes the updating of the window.</td>
</tr>
<tr>
<td>[O]</td>
<td>Opens a text file in the window.</td>
</tr>
<tr>
<td>[S]</td>
<td>Saves the contents of the window in a text file.</td>
</tr>
<tr>
<td>[P]</td>
<td>DPrints the contents of the window.</td>
</tr>
</tbody>
</table>
2.15 ANSI-C function descriptions

2.15.1 lpszPictureName

Overview

"lpszPictureName" is the name of the picture.

If you configure an action on a property or a "Mouse-click" event in WinCC, the name of the picture is provided as "lpszPictureName" in the action. The picture name has the following structure:

<BASE PICTURE NAME>.<PICTURE WINDOW NAME>:<PICTURE NAME>. ... .<Picture window name>:<Picture name>.

The "BASE PICTURE NAME" and the "PICTURE NAME" are provided without the file extension ".PDL".

This enables you to identify the object's picture path. You can also address specific picture windows, if a process picture is opened more than once for example.

Note

Do not change the text in "lpszPictureName not even using the function "strcat".

2.15.2 Standard functions

2.15.2.1 Standard functions - short description

The system provides standard functions. You can modify these functions to adapt them to your personal needs. Furthermore, you can create your own standard functions.

The basic system provides you with standard functions. They are divided into the following function groups:

- Alarm
- Graphics
- Report
- TagLog
- WinCC
- Windows

The "Obsolete functions" directory contains standard functions that were used to control the control before WinCC V7.
If the corresponding options have been installed, the following additional function groups are available:

- Options
- Split Screen Manager
- userarc (user archives)

2.15.2.2 Alarm

AcknowledgeMessage

Function

Acknowledges the message with the number that has been sent as a parameter in the message system.

Syntax

```c
void AcknowledgeMessage(DWORD MsgNr)
```

Parameters

- **MsgNo**
  - Message to be acknowledged

**Note**

Make sure a configured message exists for the transferred message number. To use the function on a client with its own project, a standard server for alarms has to be configured on the client.

GCreateMyOperationMsg

Function

The "GCreateMyOperationMsg" standard function makes it possible to trigger your own operator input message in the message system. The message with the "dwMsgNum" message number must have already been configured as the operator input message.
**Syntax**

```c
int GCreateMyOperationMsg( DWORD dwFlags, DWORD dwMsgNum, char* lpszPictureName, char* lpszObjectName, DWORD dwMyTextID, double doValueOld, double doValueNew, char* pszComment)
```

**Parameters**

**dwFlags**

The message form can be selected using the "dwFlags" parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAG_COMMENT_PARAMETER</td>
<td>0x00000001</td>
<td>The text is entered as a comment directly into the message in Runtime, without its own comment dialog. The pointer to the comment must not equal &quot;NULL.&quot;</td>
</tr>
<tr>
<td>FLAG_COMMENT_DIALOG</td>
<td>0x00000003</td>
<td>A comment dialog appears. The comment entered there is transferred to the message.</td>
</tr>
<tr>
<td>FLAG_TEXTID_PARAMETER</td>
<td>0x00000100</td>
<td>The text ID of a text from the TextLibrary is provided as the accompanying process value of the message.</td>
</tr>
</tbody>
</table>

**dwMsgNum**

WinCC message number of a self-created operator input message.

**lpszPictureName**

Pointer to the picture name of the picture from which the function is called.

**lpszObjectName**

Pointer to the WinCC tag name to which the old values and new values belong.

The name is forwarded as the instance name of the operator input message and entered in the accompanying process value "1".

**dwMyTextID**

Text ID of a text from the TextLibrary.

When the "FLAG_TEXTID_PARAMETER" is set, the text ID is provided as the numeric accompanying process value "8" of the message and is displayed as a number in process value block 8. So that the language-dependent text from the TextLibrary is displayed in the message, you must enter format statement "@8%s@" in the message text block.

**doValueOld**

Numeric old value of the WinCC tags with the name specified in "lpszObjectName". "doValueOld" is entered in the accompanying process value "2" of the message. The function itself has no option of reading a tag value before the action. For this, use the provided "GetTag..." feature.
doValueNew
Numeric new value of the WinCC tags with the name specified in "lpszObjectName". "doValueNew" is entered in the accompanying process value "3" of the message. The function itself has no option of reading a tag value after the action. For this, use the provided "GetTag..." feature.

pszComment
Comment text or empty string.

When "FLAG_COMMENT_PARAMETER" is set, the text is entered directly into the message in Runtime as a comment. The message does not need a separate comment dialog.

Return value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function has been completed without any errors.</td>
</tr>
<tr>
<td>-101</td>
<td>The message editing could not be started.</td>
</tr>
<tr>
<td>-201</td>
<td>An error occurred when calling the &quot;MSRTGetComment()&quot; feature.</td>
</tr>
<tr>
<td>-301</td>
<td>An error occurred when calling the &quot;MSRTCreateMsgInstanceWithComment()&quot; feature.</td>
</tr>
</tbody>
</table>

Note
Make sure that only operator input messages are used for the "GCreateMyOperationMsg" function. The use of messages of different message classes is not permitted.

Please note the role of the standard server when using the function with a Client. For more information see the chapter "Client configuration".

GMsgFunction

Function
This function provides the message data.

It is a global function for single messages. It is called for each message for which the "Triggers an action" parameter has been set.

Evaluation of the message data is best made in a project function called from GMsgFunction.

Syntax

BOOL GMsgFunction(char* pszMsgData)
Parameters

**pszMsgData**
Pointer to a string whose data are mapped with scanf to the MSG_RTDATA_STRUCT structure.

The "MSG_RTDATA_STRUCT" string contains the following data, which are separated from each other with ";":
1. Telegram time
2. Process values
3. Instance
4. User
5. Computer
6. Current time in format "yyyy.mm.dd, hh:mm:ss.mmm"

**Note**
- The value "Instance" of string "MSG_RTDATA_STRUCT" is only supplied if an instance message was triggered.
- The values "User" and "Computer" of the string "MSG_RTDATA_STRUCT" are only supplied if a comment was provided during the creation of the message with the same call.

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

**Note**
Please note that modified standard functions are overwritten by a WinCC installation so that the changes will be lost.
2.15.2.3 Graphics

Graphics - short description

The Graphics group contains functions for programming the graphic system.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetLinkedVariable

Function

Provides the name of the variable linked to a certain object property.

Syntax

char* GetLinkedVariable(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

Parameters

lpszPictureName
Pointer to the picture

lpszObjectName
Pointer to the object

lpszPropertyName
Pointer to the object property

Return value

Pointer to the name of the tag linked to a certain object property.

GetLocalPicture

Function

Provides a pointer to the name of the picture. The picture name is the file name without the ".PDL" extension.
Syntax

    char* GetLocalPicture(char* lpszPictureName);

Parameters

    lpszPictureName
    Pointer to the picture

Return value

    Pointer on the name of the picture.

Note

    The passed call parameter lpszPictureName must have the structure provided by the graphics system for the picture paths:

    <Basic picture name>.<Picture window name>:<Picture name>. ... .<Picture window name>[<Picture name>]

    where <Basic picture name> and <Picture name> go without the ".PDL" file extension.

Example:

    In a basic picture "AAA" there is a picture window "bbb" in which a picture "CCC" is called which itself contains a picture window "ddd" in which a picture "EEE" is called.

    Then the function call

    GetLocalPicture(lpszPictureName)

    returns the pointer to the picture name:

    "EEE" if the functions is called in the picture "EEE";

    "CCC" if the functions is called in the picture "CCC";

    "AAA" if the functions is called in the picture "AAA".

GetParentPicture

Function

    Provides a pointer to the name of the picture. The picture name is the file name without the ".PDL" extension.

Syntax

    char* GetParentPicture(char* lpszPictureName);
Parameters

\textbf{lpszPictureName} \\
Pointer to the picture

Return value

Name of the current picture if the function is called in the basic picture \\
Name path of the higher-level picture if the function is called in a picture window

\underline{Note}

The passed call parameter lpszPictureName must have the structure provided by the graphics system for the picture paths: \\
<Basic picture name>.<Picture window name>:<Picture name> ... .<Picture window name>[:<Picture name>]

where <Basic picture name> and <Picture name> go without the ".PDL" file extension.

\textbf{GetParentPictureWindow}

Function

Provides a pointer to the name of the picture window.

Syntax

\texttt{char* GetParentPictureWindow(char* lpszPictureName);}

Parameters

\textbf{lpszPictureName} \\
Pointer to the picture

Return value

Pointer to the name of the picture window if the function is called in a picture displayed in a picture window of a higher-level picture \\
Call parameter lpszPictureName unchanged if the function is called in the basic picture
Note
The passed call parameter lpszPictureName must have the structure provided by the graphics system for the picture paths:

<Basic picture name>.<Picture window name>:<Picture name>. ... .<Picture window name>[<Picture name>]

where <Basic picture name> and <Picture name> go without the ".PDL" file extension.

Example:
In a basic picture "Picture_1" there is a picture window "Picture_window_1" in which a picture "Picture_2" is called.

In the picture "Picture_2" there is a picture window "Picture_window_2" in which a picture "Picture_3" is called.

Then the function call

GetParentPictureWindow(lpszPictureName)

returns the pointer to the picture window name:

"Picture_2" if the function is called in the picture "Picture_3";
"Picture_window_1" if the function is called in the picture "Picture_2";
"Picture_1" if the function is called in the picture "Picture_1".

OpenPicture

Function
Changes the specified basic picture. On the client and in case of a picture name with server prefix a picture change is performed in the picture window.

If, for example, the picture window is located in a different picture window with a server prefix, a picture change is not performed in the picture window in which the function was called.

If multiple picture windows with server prefix are integrated in the picture and the "OpenPicture()" function calls the last picture, the picture change is carried out in the first picture window. For example, "screen1.window1(screen2.window2(screen3.window3(screen4.OpenPicture)))" executes a picture change in "window1".

WinCC Alarm Control: WinCC server or WinCC client with its own project
If you have selected a server package in the WinCC AlarmControl, the picture change takes place in the picture window of the WinCC Alarm Control when called with "Loop in Alarm".
Syntax

```c
void OpenPicture(Picture PictureName)
```

Parameters

- **Picture name**
  Picture name

Registry2

Function

This function manages a list of string pairs (String0, String1).

It knows the following types of calls controlled by the mode parameter:

- `Registry2("set", "String0", "String1");`
  Includes the passed string pair into the list.

- `Registry2("get", "String0", NULL);`
  Returns the first string pair partner String1 which belongs to the passed String0 and then deletes the string pair from the list.

- `Registry2("reset", NULL, NULL);`
  Deletes all string pairs from the list.

- `Registry2("display", NULL, NULL);`
  Shows the string pairs currently stored in the list in a Global Script diagnostics window.

Syntax

```c
char* Registry2(char* mode, char* String0, char* String1);
```

Parameters

- **mode**
  Defines the working principle of the function.

  - `set`  Incorporation of the string pair into the list
  - `get`  Determination of the first string pair partner for String0 and deletion of the string pair from the list
  - `reset`  Deletion of all string pairs
  - `display`  Display of the string pairs in a Global Script diagnostics window

- **String0**
  The parameter supply depends on the working principle of the function.
String1
The parameter supply depends on the working principle of the function.

Return value
In the mode=get mode a pointer to the first string pair partner is returned.

Note
This function is used in conjunction with the picture module technology.
If you work with the "Create faceplate as type" and "Create instance(s) in the process picture" wizards in the "Faceplates" tab of the Dynamic Wizard, using the "Registry2" function is not permitted!

2.15.2.4 Obsolete functions

Alarm

AXC_OnBtnAlarmHidingList

Function
This function displays the list of hidden messages in a message window.

Syntax
BOOL AXC_OnBtnAlarmHidingList(char* lpszPictureName, char* lpszObjectName)

Parameters
lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnArcLong

Function
This function displays the messages stored in a long-term archive list in a message window.

Syntax
BOOL AXC_OnBtnArcLong (char* lpszPictureName, char* lpszObjectName)

Parameters
lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnArcShort

Function
This function displays the messages stored in a short-term archive list in a message window.

Syntax
BOOL AXC_OnBtnArcShort(char* lpszPictureName, char* lpszObjectName)

Parameters
- lpszPictureName
  Pointer to the name of the picture in which the WinCC Alarm Control is located
- lpszObjectName
  Pointer to the object name of the WinCC Alarm Control

Return value
- TRUE
  The function has been completed without any errors.
- FALSE
  An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnComment

Function
External message window operation
This function displays the comment of the previously selected messages.

Syntax
BOOL AXC_OnBtnComment (char* lpszPictureName, char* lpszObjectName)
Parameters

- **lpszPictureName**: Pointer to the name of the picture in which the WinCC Alarm Control is located
- **lpszObjectName**: Pointer to the object name of the WinCC Alarm Control

Return value

- **TRUE**: The function has been completed without any errors.
- **FALSE**: An error has occurred.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnEmergAckn

Function

External message window operation
This function opens the acknowledgement dialog (emergency acknowledgement/reset).

Syntax

```c
BOOL AXC_OnBtnEmergAckn(char* lpszPictureName, char* lpszObjectName)
```

Parameters

- **lpszPictureName**: Pointer to the name of the picture in which the WinCC Alarm Control is located
- **lpszObjectName**: Pointer to the object name of the WinCC Alarm Control

Return value

- **TRUE**: The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnHideDlg

Function
This function opens the display options dialog for defining the messages that are to be displayed in the message window. The options are "All messages", "Shown messages" or "Hidden messages".

Syntax
BOOL AXC_OnBtnHideDlg(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnHideUnhideMsg

Function

The function hides the selected message or displays again the hidden message.

Syntax

BOOL AXC_OnBtnHideUnhideMsg(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnHit

Function

This function displays the messages stored in the hit list in a message window.

Syntax

BOOL AXC_OnBtnHit (char* lpszPictureName, char* lpszObjectName)
Parameters

**lpszPictureName**
Pointer to the name of the picture in which the WinCC Alarm Control is located

**lpszObjectName**
Pointer to the object name of the WinCC Alarm Control

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

---

**Note**
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

---

**AXC_OnBtnHornAckn**

**Function**
External message window operation
This function acknowledges the horn signal.

**Syntax**

```c
BOOL AXC_OnBtnHornAckn (char* lpszPictureName, char* lpszObjectName)
```

**Parameters**

**lpszPictureName**
Pointer to the name of the picture in which the WinCC Alarm Control is located

**lpszObjectName**
Pointer to the object name of the WinCC Alarm Control

**Return value**

**TRUE**
The function has been completed without any errors.
**AXC_OnBtnInfo**

**Function**

External message window operation
This function displays the information text.

**Syntax**

```
BOOL AXC_OnBtnInfo (char* lpszPictureName, char* lpszObjectName)
```

**Parameters**

- **lpszPictureName**
  Pointer to the name of the picture in which the WinCC Alarm Control is located

- **lpszObjectName**
  Pointer to the object name of the WinCC Alarm Control

**Return value**

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

**Note**

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnLock

Function

External message window operation
This function opens the "Set the Lock List Parameters" dialog.

Syntax

BOOL AXC_OnBtnLock (char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnLockUnlock

Function

This function locks the selected message in the message window. This message will then no longer be archived.
This function unlocks the selected message in the lock list.

Syntax

BOOL AXC_OnBtnLockUnlock (char* lpszPictureName, char* lpszObjectName)
Parameters

\textbf{lpszPictureName}

Pointer to the name of the picture in which the WinCC Alarm Control is located

\textbf{lpszObjectName}

Pointer to the object name of the WinCC Alarm Control

Return value

\textbf{TRUE}

The function has been completed without any errors.

\textbf{FALSE}

An error has occurred.

\underline{Note}

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

\textbf{AXC\_OnBtnLockWin}

Function

External message window operation.

This function calls the lock list.

Syntax

\texttt{BOOL AXC\_OnBtnLockWin (char* lpszPictureName, char* lpszObjectName)}

Parameters

\textbf{lpszPictureName}

Pointer to the name of the picture in which the WinCC Alarm Control is located

\textbf{lpszObjectName}

Pointer to the object name of the WinCC Alarm Control

Return value

\textbf{TRUE}

The function has been completed without any errors.
FALSE
An error has occurred.

---

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnLoop

Function
External message window operation
This function triggers the "LoopInAlarm" function of the selected message.

Syntax
BOOL AXC_OnBtnLoop (char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

---

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnMsgFirst

Function
External message window operation
This function switches to the beginning of the message list.

Syntax
BOOL AXC_OnBtnMsgFirst (char* lpszPictureName, char* lpszObjectName)

Parameters
lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Alarm Control as of WinCC V7.0.

AXC_OnBtnMsgLast

Function
External message window operation
This function switches to the beginning of the message list.

Syntax
BOOL AXC_OnBtnMsgLast (char* lpszPictureName, char* lpszObjectName)
Parameters

- **lpszPictureName**
  Pointer to the name of the picture in which the WinCC Alarm Control is located

- **lpszObjectName**
  Pointer to the object name of the WinCC Alarm Control

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnMsgNext

Function

External message window operation
This function switches to the next message in the message list.

Syntax

```c
BOOL AXC_OnBtnMsgNext (char* lpszPictureName, char* lpszObjectName)
```

Parameters

- **lpszPictureName**
  Pointer to the name of the picture in which the WinCC Alarm Control is located

- **lpszObjectName**
  Pointer to the object name of the WinCC Alarm Control

Return value

- **TRUE**
  The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnMsgPrev

Function
External message window operation
This function switches to the previous message in the message list.

Syntax
BOOL AXC_OnBtnMsgPrev (char* lpszPictureName, char* lpszObjectName)

Parameters
lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnMsgWin

Function

External message window operation
This function calls the message list.

Note
The message list contains the currently pending and unacknowledged messages.

Syntax

BOOL AXC_OnBtnMsgWin (char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnPrint

Function

External message window operation
All messages fulfilling the selection criterion set in the Alarm Control are output to the printer.
Syntax

BOOL AXC_OnBtnPrint(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnProtocol

Function

External message window operation

Printing of the current view of the Alarm Control is started. All messages fulfilling the selection criterion set in the Alarm Control are output to the printer.

Syntax

BOOL AXC_OnBtnProtocol(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnScroll

Function
External message window operation
This function activates or deactivates the horizontal and vertical scroll functions.

Syntax
BOOL AXC_OnBtnScroll(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnSelect

Function

External message window operation
This function opens the "Specify Selection" dialog for the displayed list.

Syntax

BOOL AXC_OnBtnSelect(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnSinglAckn

Function

External message window operation
This function acknowledges the currently selected message.

Syntax

BOOL AXC_OnBtnSinglAckn(char* lpszPictureName, char* lpszObjectName)
Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnSortDlg

Function
External operation of the message window
This function opens the dialog for setting a user-defined sorting of the displayed messages for the displayed list.

Syntax

BOOL AXC_OnBtnSortDlg(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_OnBtnTimeBase

Function
External operation of the message window
This function opens the dialog for setting the time base for the times shown in the messages.

Syntax

BOOL AXC_OnBtnTimeBase(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.
AXC_OnBtnVisibleAckn

Function

External message window operation
All visible messages in the message window are acknowledged (group acknowledgement).

Syntax

BOOL AXC_OnBtnVisibleAckn(char* lpszPictureName, char* lpszObjectName)

Parameters

lpszPictureName
Pointer to the name of the picture in which the WinCC Alarm Control is located

lpszObjectName
Pointer to the object name of the WinCC Alarm Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

AXC_SetFilter

Function

External message window operation
This function sets a filter for WinCC Alarm Control to show a portion of the existing messages according to the filter criterion.

Syntax

BOOL AXC_SetFilter(char* lpszPictureName, char* lpszObjectName, LPMSG_FILTER_STRUCT lpMsgFilter, LPCMN_ERROR lpError)
Parameters

`lpszPictureName`
 Pointer to the name of the picture in which the WinCC Alarm Control is located

`lpszObjectName`
 Pointer to the WinCC Alarm Control name

`lpMsgFilter`
 Pointer to the structure containing the filter criterion

`lpError`
 Pointer to the structure of the error description

Return value

`TRUE`
 The function has been completed without any errors.

`FALSE`
 An error has occurred.

Note
The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

Report

ReportJob

Function
Depending on the value of the `IpMethodName` parameter a print job or the preview for a print job is started.

Syntax

```c
void ReportJob(LPSTR lpJobName, LPSTR lpMethodName)
```

Parameters

`lpJobName`
 Pointer to the name of the print job
lpMethodName

PRINTJOB Print job is started
PREVIEW Preview of the print job is started

Note
This function is replaced by the RPTJobPreview and RPTJobPrint functions and should no longer be used.

TagLog

TOOLBAR_BUTTONS

TlgTableWindowPressEditRecordButton

Function
The editing of the table window is blocked or enabled (toggle function).
If editing is enabled, updating of the table window is stopped at the same time.
The updating of the table window remains to be stopped afterward, even if editing is blocked by a further function call.

Syntax
BOOL TlgTableWindowPressEditRecordButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.

TlgTableWindowPressFirstButton

Function
Displays the first data records of the display area in the table window.
The number of displayed data records depends on the configured time range.

Syntax
BOOL TlgTableWindowPressFirstButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

TlgTableWindowPressHelpButton

Function
Displays the online help for the table window.
Syntax

BOOL TlgTableWindowPressHelpButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressInsertRecordButton

Syntax

BOOL TlgTableWindowPressInsertRecordButton(char* lpszWindowName)

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressLastButton

Function
Displays the last data records of the display area in the table window.
The number of displayed data records depends on the configured time range.
Syntax

BOOL TlgTableWindowPressLastButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressNextButton

Function
The data records following the current display area are displayed in the table window. The number of displayed data records depends on the configured time range.

Syntax

BOOL TlgTableWindowPressNextButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.

TlgTableWindowPressNextItemButton

Function
The columns of the table window are moved one column to the left, the left column taking the position of the right column.

Syntax
BOOL TlgTableWindowPressNextItemButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.

TlgTableWindowPressOpenArchiveVariableSelectionDlgButton

Function
Opens the dialog for connecting table columns to archives and tags.
Syntax

BOOL TlgTableWindowPressOpenArchiveVariableSelectionDlgButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressOpenDlgButton

Function

Opens the dialog for online configuration of the table window.

Syntax

BOOL TlgTableWindowPressOpenDlgButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressOpenItemSelectDlgButton

Function
Opens the dialog for selecting the visible columns and the first column of the table window.

Syntax
BOOL TlgTableWindowPressOpenItemSelectDlgButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressOpenTimeSelectDlgButton

Function
Opens the dialog for setting the time range to be displayed in the table columns.
Syntax

BOOL TlgTableWindowPressOpenTimeSelectDlgButton(char* lpszWindowNumber)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressPrevButton

Function

The data records preceding the current display area are displayed in the table window.
The number of displayed data records depends on the configured time range.

Syntax

BOOL TlgTableWindowPressPrevButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.

TlgTableWindowPressPrevItemButton

Function
The columns of the table window are moved one column to the right, the right column taking the position of the left column.

Syntax
BOOL TlgTableWindowPressPrevItemButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.
TlgTableWindowPressRemoveRecordButton

Syntax

BOOL TlgTableWindowPressRemoveRecordButton(char* lpszWindowName)

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTableWindowPressStartStopButton

Function

Updating of the table window is switched on or off (toggle function).

Syntax

BOOL TlgTableWindowPressStartStopButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Table Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTrendWindowPressFirstButton

Function

Displays the first data records of the display area in the trend window.
The number of displayed data records depends on the configured time range.

Syntax

BOOL TlgTrendWindowPressFirstButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online TrendControl as of WinCC V7.0.

TlgTrendWindowPressHelpButton

Function
Displays the online help for the trend window.

Syntax

BOOL TlgTableWindowPressNextButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
**TlgTrendWindowPressLastButton**

**Function**
Displays the last data records of the display area in the trend window.
The number of displayed data records depends on the configured time range.

**Syntax**

```c
BOOL TlgTrendWindowPressLastButton(char* lpszWindowName)
```

**Parameter**

- **lpszWindowName**
  Pointer to the window title of the WinCC Online Trend Control

**Return value**

- **TRUE**
The function has been completed without any errors.
- **FALSE**
  An error has occurred.

**Note**
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

---

**TlgTrendWindowPressLinealButton**

**Function**
The ruler of the trend window is shown or hidden (toggle function).
The ruler can be moved by means of the "cursor left" and "cursor right" buttons.
**Syntax**

```c
BOOL TlgTableWindowPressNextButton(char* lpszWindowName)
```

**Parameter**

- **lpszWindowName**
  Pointer to the window title of the WinCC Online Trend Control

**Return value**

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

**Note**

The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

---

**TlgTrendWindowPressNextButton**

**Function**

The data records following the current display area are displayed in the trend window. The number of displayed data records depends on the configured time range.

**Syntax**

```c
BOOL TlgTrendWindowPressNextButton(char* lpszWindowName)
```

**Parameter**

- **lpszWindowName**
  Pointer to the window title of the WinCC Online Trend Control

**Return value**

- **TRUE**
  The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

_TlgTrendWindowPressNextItemButton_

**Function**
Brings all trends in the trend window one layer to the front.
The trend in the foreground is moved into the background.

**Syntax**
BOOL _TlgTrendWindowPressNextItemButton_(char* lpszWindowName)

**Parameter**

*lpszWindowName*
Pointer to the window title of the WinCC Online Trend Control

**Return value**

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

_TlgTrendWindowPressOneToOneButton_

**Function**
Restores the standard size (1:1) in the trend window.
Syntax

BOOL TlgTrendWindowPressOneToOneButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressOpenArchiveVariableSelectionDlgButton

Function

Opens the dialog for connecting trends to archives and tags.

Syntax

BOOL TlgTrendWindowPressOpenArchiveVariableSelectionDlgButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
TlgTrendWindowPressOpenDlgButton

Function
Opens the dialog for online configuration of the trend window.

Syntax
BOOL TlgTrendWindowPressOpenDlgButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressOpenItemSelectDlgButton

Function
Opens the dialog for selecting the visible trends and the trend which is to be in the foreground.
Syntax

BOOL TlgTrendWindowPressOpenItemSelectDlgButton(char* lpszWindowNumber)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressOpenTimeSelectDlgButton

Function
Opens the dialog for setting the time range to be displayed.

Syntax

BOOL TlgTrendWindowPressOpenTimeSelectDlgButton(char* lpszWindowNumber)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

TlgTrendWindowPressPrevButton

Function
The data records preceding the current display area are displayed in the trend window. The number of displayed data records depends on the configured time range.

Syntax
BOOL TlgTrendWindowPressPrevButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

TlgTrendWindowPressPrevItemButton

Function
Brings all trends in the trend window one layer to the back. The trend in the background is moved to the foreground.
Syntax

BOOL TlgTrendWindowPressPrevItemButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressPrintButton

Function
The current view of the trends is output in accordance with the display configured for the WinCC Trend Control.

Syntax

BOOL TlgTrendWindowPressPrintButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressReportSaveButton

Function
The displayed trend window data is saved in a text file.

Syntax
BOOL TlgTrendWindowPressReportSaveButton (char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressStartStopButton

Function
Updating of the trend window is switched on or off (toggle function).
Syntax

BOOL TlgTrendWindowPressStartStopButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

TlgTrendWindowPressStatsResultButton

Function

Starts the evaluation of data in the selected time area.
The statistic values minimum, maximum, average and standard deviation are calculated.

Syntax

BOOL TlgTrendWindowPressStatsResultButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressStatsSelectRangeButton

Function
To select the time range for the statistics function, the rulers for start and end time are displayed.

Syntax
BOOL TlgTrendWindowPressStatsSelectRangeButton(char* lpszWindowName)

Parameter
lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressZoomInButton

Function
The zoom in the trend window is activated. The zoom range can only be selected with the mouse.
Syntax

BOOL TlgTrendWindowPressZoomInButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

TlgTrendWindowPressZoomOutButton

Function

The trend window is restored to the state in which it was before the zoom was activated. The zoom is deactivated.

The zoom range can only be selected with the mouse (also see TlgTrendWindowPressZoomInButton).

Syntax

BOOL TlgTrendWindowPressZoomOutButton(char* lpszWindowName)

Parameter

lpszWindowName
Pointer to the window title of the WinCC Online Trend Control

Return value

TRUE
The function has been completed without any errors.
An error has occurred.

Note
The standard function is no longer supported for the new WinCC Online Trend Control as of WinCC V7.0.

Template

TlgGetNumberOfColumns

Function
Provides the number of columns in the table window.
The window title of the corresponding WinCC Online Table Control is passed with the lpszTemplate parameter.

Syntax
int TlgGetNumberOfColumns(char* lpszTemplate)

Parameter

lpszTemplate
Pointer to the window title of the WinCC Online Table Control

Return value
Number of columns in a table window

Note
The standard function is no longer supported for the new WinCC Online Table Control as of WinCC V7.0.

TlgGetNumberOfRows

Function
Provides the number of lines in the table window.
The window title of the corresponding WinCC Online Table Control is passed with the lpszTemplate parameter.
Syntax

```c
int TlgGetNumberOfRows(char* lpszTemplate)
```

Parameter

**lpszTemplate**
Pointer to the window title of the WinCC Online Table Control

Return value

Number of lines in the table window

---

**Note**
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

---

**TlgGetNumberOfTrends**

Function

Provides the number of trends in the trend window.

The window title of the corresponding WinCC Online Trend Control is passed with the `lpszTemplate` parameter.

Syntax

```c
int TlgGetNumberOfTrends(char* lpszTemplate)
```

Parameter

**lpszTemplate**
Pointer to the window title of the WinCC Online Trend Control

Return value

Number of trends in the trend window

---

**Note**
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.
TlgGetRowPosition

Function

Provides the current position of the line pointer in the table window.
The window title of the corresponding WinCC Online Table Control is passed with the lpszTemplate parameter.

Syntax

```c
int TlgGetRowPosition(char* lpszTemplate)
```

Parameter

**lpszTemplate**
Pointer to the window title of the WinCC Online Table Control

Return value

Current position of the line pointer in the table window

**Note**
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgGetRulerArchivNameTrend

Function

Provides the archive name of the trend with the nTrend number in the trend window at the ruler position.
The window title of the corresponding WinCC Online Trend Control is passed with the lpszTemplate parameter.

Syntax

```c
char* TlgGetRulerArchivNameTrend(char* lpszTemplate, int nTrend)
```

Parameter

**lpszTemplate**
Pointer to the window title of the WinCC Online Trend Control
**nTrend**
Number of the trend
(0 <= nTrend <= Number of visible trends - 1)

**Return value**
Archive name of the trend with the nTrend number in the trend window at the ruler position

**Note**
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

---

**TlgGetRulerTimeTrend**

**Function**
Provides the time of the trend with the nTrend number in the trend window at the ruler position.
The window title of the corresponding WinCC Online Trend Control is passed with the lpszTemplate parameter.

**Syntax**
 SYSTEMTIME TlgGetRulerTimeTrend(char* lpszTemplate, int nTrend)

**Parameter**

**lpszTemplate**
Pointer to the window title of the WinCC Online Trend Control

**nTrend**
Number of the trend
(0 <= nTrend <= Number of visible trends - 1)

**Return value**
Time of the trend with the nTrend number in the trend window at the ruler position

**Note**
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.
See also

TlgGetRulerTimeTrend example (Page 1587)

TlgGetRulerValueTrend

Function

Provides the value of the trend with the nTrend number in the trend window at the ruler position. The window title of the corresponding WinCC Online Trend Control is passed with the lpszTemplate parameter.

Syntax

double TlgGetRulerValueTrend(char* lpszTemplate, int nTrend)

Parameter

lpszTemplate
Pointer to the window title of the WinCC Online Trend Control

nTrend
Number of the trend
(0 <= nTrend <= Number of visible trends - 1)

Return value

Value of the trend with the nTrend number in the trend window at the ruler position

Note

The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

TlgGetRulerVariableNameTrend

Function

Provides the tag name of the trend with the nTrend number in the trend window. The window title of the corresponding WinCC Online Trend Control is passed with the lpszTemplate parameter.
Syntax

```c
char* TlgGetRulerVariableNameTrend(char* lpszTemplate, int nTrend)
```

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lpszTemplate</code></td>
<td>Pointer to the window title of the WinCC Online Trend Control</td>
</tr>
<tr>
<td><code>nTrend</code></td>
<td>Number of the trend</td>
</tr>
<tr>
<td></td>
<td>(0 &lt;= nTrend &lt;= Number of visible trends - 1)</td>
</tr>
</tbody>
</table>

Return value

The tag name of the trend with the nTrend number in the trend window.

Note

The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

TlgGetTextAtPos

Function

Provides the content of a cell of the table window as text for process value archives and user archives.

The cell is specified by nColumn and nLine.

The window title of the corresponding WinCC Online Table Control is passed with the lpszTemplate parameter.

Syntax

```c
char* TlgGetTextAtPos(char* lpszTemplate, int nColumn, int nLine)
```

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lpszTemplate</code></td>
<td>Pointer to the window title of the WinCC Online Table Control</td>
</tr>
<tr>
<td><code>nColumn</code></td>
<td>Number of the column</td>
</tr>
<tr>
<td><code>nLine</code></td>
<td>Number of the line</td>
</tr>
</tbody>
</table>
nLine
Number of the line

Return value
Content of the cell of a table window as text

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgGetColumnPosition

Function
Provides the current position of the column pointer in the table window as column index.

Syntax
int TlgGetColumnPosition( char* lpszTemplate )

Parameter
lpszTemplate
Pointer to the window title of the WinCC Online Table Control

Return value
Current position of the column pointer in a table window

Note
The standard function is no longer supported for the new WinCC OnlineTableControl as of WinCC V7.0.

TlgTrendWindowActivateCurve

Function
Activates a certain trend in WinCC Online Trend Control via the configured name of the trend. This function is executed independently of the visibility or foreground position of the trend.
Syntax

BOOL TlgTrendWindowActivateCurve(char* lpszPictureName, char* lpszObjectName, char* szValue)

Parameter

lpszPictureName
Picture name

lpszObjectName
Name of Trend Control

szValue
Name of the curve

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The standard function is no longer supported for the new WinCC OnlineTrendControl as of WinCC V7.0.

2.15.2.5 Report

Report - short description

The Report group contains functions with which to start the print preview of a print job or the printout itself.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
RPTJobPreview

Function
The preview of a print job is started.

Syntax
BOOL RPTJobPreview(LPSTR lpJobName)

Parameters
lpJobName
Pointer to the name of the print job

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

RPTJobPrint

Function
A print job is started.

Syntax
BOOL RPTJobPrint(LPSTR lpJobName)

Parameters
lpJobName
Pointer to the name of the print job

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

RptShowError

Function
This function provides an error description for a failed print job.
The function is already integrated into the RptJobPrint and RptJobPreview standard functions and does not have to be called separately.
The error description is displayed in a Global Script diagnostics window.

Note
As RptShowError is a standard function the output type and form can be changed if required.
Please note that modified standard functions are overwritten by a WinCC installation so that the changes will be lost.

Syntax
void RptShowError ( LPCSTR pszFailedFunction, CMN_ERRORA* pCmnErrorA )

Parameters
pszFailedFunction
Pointer to the name of the failed function.
If this pointer is NULL there will be no output of the function name.

pCmnErrorA
Pointer to the error structure of the failed function.
If this pointer is NULL there will be no output of the error structure.
STRUCutunes_TABLES_ERROR_STRUCTURE
WinCC - short description

The WinCC group contains functions which affect the entire WinCC system.

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetHWDiag

**Function**

This function realizes the direct start of diagnosis at runtime triggered by an event, which must be configured, exercised on an object.

If the event occurs, the hardware diagnostics function is started from STEP7 for the associated PLC.

The following conditions must be fulfilled in order to use the function:

- The WinCC project, with the picture from which access should occur, and the STEP7 project must be on the same computer.
- The WinCC project must be stored as a subdirectory of the STEP7 project (STEP7 Projekt \wincproj\WinCC Projekt).
- The S7 tags have been mapped to WinCC.

**Syntax**

```c
BOOL GetHWDiag(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpProperties)
```

**Parameters**

- **lpszPictureName**
  
  Name of the picture (PDL file) that contains the tag that will be used for the entry point for the hardware diagnostics

  Since the name "lpszPictureName" stands for the current picture, entries are only required here in cases where it is necessary to access an object tag in a different picture.

- **lpszObjectName**

  Name of the object in the picture that connected with the tag that will be used for the entry point for the hardware diagnostics
Since the name "lpszObjectName" stands for the current object entries are only required here in cases where it is necessary to access a tag in a different object.

lpProperties
Name of the attribute that is connected with the tag that will be used for the entry point for the hardware diagnostics
If multiple attribute are entered, they must be separated by semicolons (";").

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

GetHWDiagLevel

Function
Checks the logged-in user’s authorization on the basis of the User Administrator function number in dwLevel.
Then, diagnostics is started directly during runtime and is triggered by an event, which has to be configured, occurring on an object.
If the event occurs, the hardware diagnostics function is started from STEP7 for the associated PLC.
The following conditions must be fulfilled in order to use the function:

● The WinCC project, with the picture from which access should occur, and the STEP7 project must be on the same computer.
● The WinCC project must be stored as a subdirectory of the STEP7 project (STEP7 Projekt \wincproj\WinCC Projekt).
● The S7 tags have been mapped to WinCC.
● In order for the user logged into WinCC to edit the hardware diagnostics dialog, the user must have a WinCC user authorization matching the number passed by the function call in the parameter "dwLevel".

Syntax

BOOL GetHWDiagLevel(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpProperties, DWORD dwLevel)
Parameters

**lpszPictureName**
Name of the picture (PDL file) that contains the tag that will be used for the entry point for the hardware diagnostics

Since the name "lpszPictureName" stands for the current picture, entries are only required here in cases where it is necessary to access an object tag in a different picture.

**lpszObjectName**
Name of the object in the picture that connected with the tag that will be used for the entry point for the hardware diagnostics

Since the name "lpszObjectName" stands for the current object entries are only required here in cases where it is necessary to access a tag in a different object.

**lpProperties**
Name of the attribute that is connected with the tag that will be used for the entry point for the hardware diagnostics

If multiple attribute are entered, they must be separated by semicolons (";").

**dwLevel**
Level number for STEP7 write permissions.

This can be defined in User Administrator.

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

GetKopFupAwl

Function

This function performs the network entry jump of WinCC into the STEP7 Editor "KFA".

When executing this function two tasks are performed:

- Determination of the required date for the network entry jump from WinCC.
- Transfer of the data to Step7 and finding the places of use of the operand in a STEP7 program by means of AUTAPI.
Syntax

BOOL GetKopFupAwl(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpProperties)

Parameters

lpszPictureName
Name of the picture (PDL file) that contains the tag that will be used for the network entry jump
Since the name "lpszPictureName" stands for the current picture, entries are only required here in cases where it is necessary to access an object tag in a different picture.

lpszObjectName
Name of the object in the picture that connected with the tag that will be used for the network entry jump
Since the name "lpszObjectName" stands for the current object entries are only required here in cases where it is necessary to access a tag in a different object.

IpProperties
Name of the attribute that is connected with the tag that will be used for the network entry jump
If multiple attribute are entered, they must be separated by semicolons (";").

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

GetKopFupAwlLevel

Function
Checks the active user's authorization on the basis of the User Administrator function number in dwLevel and then performs the entry jump into the STEP7 Editor "KFA".
When executing this function three tasks are performed:

- Determination of the required date for the network entry jump from WinCC.
- Authorization check for the active user within WinCC.
- Transfer of the data to STEP7 and finding the places of use of the operand in a STEP7 program by means of AUTAPI.

**Note**

Depending on the result of the authorization check in WinCC the user has either only reading rights in STEP7 or is authorized to change S7 data.

**Syntax**

```c
BOOL GetKopFupAwlLevel(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpProperties, DWORD dwLevel)
```

**Parameters**

- **lpszPictureName**
  
  Name of the picture (PDL file) that contains the tag that will be used for the network entry jump
  
  Since the name "lpszPictureName" stands for the current picture, entries are only required here in cases where it is necessary to access an object tag in a different picture.

- **lpszObjectName**
  
  Name of the object in the picture that connected with the tag that will be used for the network entry jump
  
  Since the name "lpszObjectName" stands for the current object entries are only required here in cases where it is necessary to access a tag in a different object.

- **lpProperties**
  
  Name of the attribute that is connected with the tag that will be used for the network entry jump
  
  If multiple attribute are entered, they must be separated by semicolons (";").

- **dwLevel**
  
  Level number for STEP7 write permissions.
  
  This can be defined in User Administrator.

**Return value**

- **TRUE**
  
  The function has been completed without any errors.
FALSE
An error has occurred.

**OnDeactivateExecute**

**Function**
This function is called when terminating WinCC Runtime.
As this is a standard function, you can insert instructions which are then executed.

**Note**
Concerning the instructions it must be taken into account that the Runtime is terminating and therefore not all functionalities are available.
Please note that modified standard functions are overwritten by a WinCC installation so that the changes will be lost.

**Syntax**
void OnDeactivateExecute()

**OnErrorExecute**

**Function**
OnErrorExecute is called by the system when an error occurred upon executing an action or a function.
This allows you to determine the precise error cause.
The function is called by the system and does not require an additional call.
As this function is available as a standard function the output type and form can be changed if required.

**Note**
Please note that modified standard functions are overwritten by a new installation so that the changes will be lost.

**Syntax**
void OnErrorExecute(CCAPErrorExecute ErrorExecute)
Parameters

**ErrorExecute**
Structure informing about the error that has occurred

Diagnostic information

These information are displayed in a Global Script diagnostics window.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemTime</td>
<td>Time (UTC) at which the error occurred</td>
</tr>
<tr>
<td>dwErrorCode1</td>
<td>The error codes and their meaning are to be found in the structure definition</td>
</tr>
<tr>
<td>dwErrorCode2</td>
<td>The error codes and their meaning are to be found in the structure definition</td>
</tr>
<tr>
<td>szErrorText</td>
<td>Text description of the error cause</td>
</tr>
<tr>
<td>bCycle</td>
<td>Cycle type</td>
</tr>
<tr>
<td>szApplicationName</td>
<td>Error-triggering application</td>
</tr>
<tr>
<td>szFunctionName</td>
<td>FunctionID</td>
</tr>
<tr>
<td>szTagName</td>
<td>Tag name</td>
</tr>
<tr>
<td>dwCycle</td>
<td>Cycle type</td>
</tr>
<tr>
<td>szErrorTextTagName</td>
<td>Text description of the tag status</td>
</tr>
<tr>
<td>status</td>
<td>Tag status</td>
</tr>
<tr>
<td>lpszPictureName</td>
<td>Picture in which the error occurred</td>
</tr>
<tr>
<td>lpszObjectName</td>
<td>Object in which the error occurred</td>
</tr>
<tr>
<td>lpszPropertyName</td>
<td>Object property in which the error occurred</td>
</tr>
<tr>
<td>dwParamSize</td>
<td>only used internally</td>
</tr>
<tr>
<td>szErrorText</td>
<td>Text description of the error cause returned by the error structure</td>
</tr>
<tr>
<td>&quot;pError&quot;</td>
<td></td>
</tr>
</tbody>
</table>

OnTime

Function

OnTime is exclusively called by the system. The function returns the runtime of all actions or determines the actions running longer than the specified time. Time measurement can be enabled/disabled via APDIAG.

As this function is available as a standard function the output type can be influenced by changing the function code.

Note

Please note that modified standard functions are overwritten by a WinCC installation so that the changes will be lost.
Syntax
void OnTime(CCAPTime time)

Parameters

  time
  Result structure
  STRUCTURES_TABLES_CCAPTIME

2.15.2.7 Windows

Windows - short description
The Windows group contains the ProgramExecute function.
This function can be used to execute any program.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

ProgramExecute

Function
Starts the program with the specified name.

Syntax
unsigned int ProgramExecute(char* Program_Name)

Parameters

  Program_Name
  Pointer to the program name

Return value
If the return value is greater than 31, the function has been completed without any errors.
In case of an error, the return value contains one of the following error codes:

0  out of memory
2  Specified file could not be found.
3  Specified path could not be found.
11 Program could not be started.

2.15.3  Internal functions

2.15.3.1  Internal functions - short description

Internal functions are used to make graphic objects and archives dynamic and in project functions, standard functions and global script actions.

Internal functions are recognized throughout a project.

They can be neither be newly created nor can existing internal functions be modified.

Internal functions are divided into the following groups:

allocate
Functions to reserve and release working memory space

c_bib
Functions from the standard C-library

graphics
Functions to read and set properties of graphical objects

tag
Functions to read and write tags

wincc
Functions for changing languages, deactivating Runtime and ending WinCC

2.15.3.2  allocate

SysFree

Function

Releases the memory area previously reserved with the SysMalloc function.
**Syntax**

```c
void SysFree(void* lpFree);
```

**Parameters**

- `lpFree`
  Pointer to the memory area reserved with the SysMalloc function

**SysMalloc**

**Function**

Reserves memory space for an action. The memory area is assigned to the action. When the action has been completed and the result transferred, the system releases the memory again. The SysFree function can be used to release reserved memory space.

**Syntax**

```c
void* SysMalloc(unsigned long int size);
```

**Parameters**

- `size`
  Size of the memory area in bytes.

---

**2.15.3.3 c_bib**

**c_bib - short description**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.

The C-library functions printf(), printf(), fprintf() can only process 360 characters in WinCC.

---

**ctype**

**isalnum**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

**isalpha**
The function group c_bib contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**islower**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**isprint**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
ispunct

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

isspace

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

`isupper`

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

### isxdigit

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
tolower

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

toupper

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**math**

**acos**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**asin**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
atan

The function group c_bib contains C functions from the C library and is divided into:
• ctype
• math
• memory
• stdio
• stdlib
• string
• time

stdio itself is further divided into:
• char io
• directio
• error
• file
• file_pos
• output

You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

atan2

The function group c_bib contains C functions from the C library and is divided into:
• ctype
• math
• memory
• stdio
• stdlib
• string
• time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.
- The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**ceil**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**cos**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
cosh

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

exp

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function `localtime` reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.
- The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**fabs**

The function group `c_bib` contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**floor**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
fmod

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

frexp

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**ldexp**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**log**

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
**log10**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**modf**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**pow**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**sin**

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
sinh

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

sqrt

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**tan**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**tanh**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**memcpy**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**memmove**

The function group c_bib contains C functions from the C library and is divided into:
- **ctype**
- **math**
- **memory**
- **stdio**
- **stdlib**
- **string**
- **time**

stdio itself is further divided into:
- **char_io**
- **directio**
- **error**
- **file**
- **file_pos**
- **output**

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

stdio

char_io

fgetc

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**fgets**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.
- The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**fputc**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.
- The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

putc

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**ungetc**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
Directio

fwrite

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

fwrite

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**Error**

**clearerr**

The function group c_bib contains C functions from the C library and is divided into:

- cfunc
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**feof**

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

File

**fclose**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.
- The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**fflush**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**fopen**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
freopen

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

remove

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

rename

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**setbuf**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

### tmpnam

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

**File_pos**

**fgetpos**

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

The function `fsetpos` does not seem to have a specific note or description provided in the given text.
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**ftell**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**rewind**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

### Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**stdlib**

**abs**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**atof**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
吃到

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

atol

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**bsearch**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

calloc

The function group c_bib contains C functions from the C library and is divided into:
  • ctype
  • math
  • memory
  • stdio
  • stdlib
  • string
  • time
stdio itself is further divided into:
  • char_io
  • directio
  • error
  • file
  • file_pos
  • output
You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

getenv

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**labs**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
Idiv

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

malloc

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**qsort**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**rand**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
realloc

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

Note

The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

srand

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**strtod**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**strtol**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**strchr**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

**strcmp**

The function group `c_bib` contains C functions from the C library and is divided into:
- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:
- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

**strerror**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

`strlen`

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
strncat

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

strncmp

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**strncpy**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

---

### strpbrk

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

---

**Note**

The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

### Note

The function localtime reacts as follows in respect of date output:

- Numbering of the months begins with 0.
- The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

---

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**strstr**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

**strtok**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
time

asctime

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

clock

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**
The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

c**time**

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

difftime

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note
The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.
gmtime

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

---

**Note**

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.

The years are counted from 1900, beginning with 0.

The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

---

localtime

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time
stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

**Note**

The function localtime reacts as follows in respect of date output:
Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

**mktime**

The function group c_bib contains C functions from the C library and is divided into:
- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:
- char_io
- directio
- error
- file
- file_pos
- output
You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.

**strftime**

The function group `c_bib` contains C functions from the C library and is divided into:

- `ctype`
- `math`
- `memory`
- `stdio`
- `stdlib`
- `string`
- `time`

`stdio` itself is further divided into:

- `char_io`
- `directio`
- `error`
- `file`
- `file_pos`
- `output`

You can find a description of this function in related technical literature.

**Note**
The function `localtime` reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions `printf()`, `sprintf()`, `fprintf()` can only process 360 characters in WinCC.
time

The function group c_bib contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is further divided into:

- char_io
- directio
- error
- file
- file_pos
- output

You can find a description of this function in related technical literature.

Note

The function localtime reacts as follows in respect of date output:

Numbering of the months begins with 0.
The years are counted from 1900, beginning with 0.
The C-library functions printf(), sprintf(), fprintf() can only process 360 characters in WinCC.

2.15.3.4 graphics

Graphics - short description

The functions of the Graphics group allow to modify or query graphical properties of WinCC objects.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
Note
If the function is called for the picture object, set the parameter lpszObjectName = ZERO.

get

axes

GetAlignment

Function
When using bar objects, it indicates whether the text is to the right or left of the bar.

Syntax
BOOL GetAlignment(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Text is to the right of the bar

FALSE
Text is to the left of the bar

See also
Beispiel GetScaling (Page 1536)
GetAxisSection

Function
When using bar objects, it specifies the difference between the values of two neighboring axis labels.

Syntax
double GetAxisSection(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value
Difference between the values of two neighboring axis labels

GetExponent

Function
When using bar objects, it specifies whether the axis label corresponds to the decimal or exponential form.

Syntax
BOOL GetExponent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name
Return value

TRUE
Axis label in exponential form

FALSE
Axis label in decimal form

See also

Beispiel GetScaling (Page 1536)

GetLeftComma

Function
When using bar objects, it specifies the number of integers in the axis label.

Syntax

long int GetLeftComma(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
The number of integers in the axis label

GetLongStrokesBold

Function
When using bar objects, it specifies whether the main division lines on the scale are bold or regular.

Syntax

BOOL GetLongStrokesBold(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Function

GetLongStrokesOnly

When using bar objects, it specifies whether intermediate division lines are used on the scale.

Syntax

BOOL GetLongStrokesOnly(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Only main division lines are used on the bar graph scale.
FALSE
Both main and intermediate division lines are used on the bar graph scale.

See also
Beispiel GetScaling (Page 1536)

GetLongStrokesSize

Function
When using bar objects, it specifies the length of the main division lines.

Syntax
long int GetLongStrokesSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Length of the main division lines as numeric value

GetLongStrokesTextEach

Function
When using bar objects, it specifies the interval between the main division lines being assigned a label.

Syntax
long int GetLongStrokesTextEach(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name
IpszObjectName
Object name

Return value
Label of the main division lines as numeric value

Example:
Return value = 1 -> Every main division line is assigned a label.
Return value = 2 -> Every 2nd main division line is assigned a label.
etc.

GetRightComma

Function
When using bar objects, it specifies the number of decimal places in the axis label.

Syntax
`long int GetRightComma(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);`

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
The number of decimal places in the axis label

GetScaleTicks

Function
When using bar objects, it specifies the scale marks as number of scale sections. A scale section is a part of the scale bounded by two main tick marks.
Syntax

long int GetScaleTicks(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Scale marks as number of scale sections

Note
The number of scale sections is given as 0, if the bar object itself calculates a suitable scale unit.

GetScaling

Function
When using bar objects, it specifies whether the scale is activated or deactivated.

Syntax

BOOL GetScaling(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Display with scale
FALSE
Display without scale

See also
Beispiel GetScaling (Page 1536)

GetScalingType

Function
When using bar objects, it specifies the type of bar scaling.

Syntax
long int GetScalingType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Type of bar scaling as numeric value

See also
Bar Scaling (Page 1590)
**color**

**Color - short description**

The various color properties of objects can be modified or queried using the functions in the Color group.

---

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

---

**GetBackColor**

**Function**

Specifies the background color of the object as a numeric value.

**Syntax**

```
long int GetBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

Background color of the object as a numeric value

---

**Note**

If the function is called in relation to the entire picture, set the parameter `lpszObjectName = NULL`.

---

**See also**

- GetBackColor example (Page 1518)
- Color chart (Page 1592)
GetBackColor2

Function
When using bar objects, it specifies the color of the bar as a numeric value.

Syntax
long int GetBackColor2(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value
Numeric value defining the bar color

See also
- GetBackColor example (Page 1518)
- Color chart (Page 1592)

GetBackColor3

Function
When using bar objects, it specifies the background color of the bar as a numeric value.

Syntax
long int GetBackColor3(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
Return value

Numeric value defining the bar background color

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetBackColorBottom

Function

Specifies the background color of the slider objects at the bottom right.

Syntax

long int GetBackColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the background color of the slider objects at the bottom right

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetBackColorTop

Function

Specifies the background color of the slider objects at the top left.

Syntax

long int GetBackColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

Return value

Numeric value defining the background color of the slider objects at the top left

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

GetBorderBackColor

Function

Specifies the background color of the lines or borders.

Syntax

```c
long int GetBorderBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

Return value

Numeric value defining the background color of the lines or borders

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)
GetBorderColor

Function

Specifies the line or border color as a numeric value.

Syntax

```c
long int GetBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value

Numeric value defining the color of lines or borders

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

GetBorderColorBottom

Function

Specifies the 3D border color at the bottom.

Syntax

```c
long int GetBorderColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
Return value

Numeric value defining the 3D border color at the bottom

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetBorderColorTop

Function

Specifies the 3D border color at the top.

Syntax

long int GetBorderColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

Return value

Numeric value defining the 3D border color at the top

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetButtonColor

Function

Specifies the button color of slider objects.

Syntax

long int GetButtonColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
### Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

### Return value

Numeric value defining the button color of slider objects

### See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

### GetColorBottom

**Function**

When using slider objects, it specifies the color of the bottom limit.

**Syntax**

```c
long int GetColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

Numeric value defining the color of the bottom limit of slider objects

**See also**

- GetBackColor example (Page 1518)
- Color chart (Page 1592)
GetColorTop

Function

When using slider objects, it specifies the color of the top limit.

Syntax

long int GetColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name

Return value

Numeric value defining the color of the top limit of slider objects

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

GetFillColor

Function

Specifies the color of the fill pattern.

Syntax

long int GetFillColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
GetForeColor

Function

Specifies the color of the font.

Syntax

long int GetForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the font color

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)
GetGridColor

Function

Specifies the grid color of Graphics Designer.

Syntax

long int GetGridColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the grid color of Graphics Designer

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetItemBorderBackColor

Function

Specifies the background color of the dividing line for the "text list" object.

Syntax

long int GetItemBorderBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

Numeric value defining the background color of the dividing line for the "text list" object

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetItemBorderColor

Function

Specifies the color of the dividing line for the "text list" object.

Syntax

long int GetItemBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the dividing line color for the "text list" object

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetScaleColor

Function

Specifies the scale color for bar objects.

Syntax

long int GetScaleColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

\texttt{lpzPictureName}
Picture name

\texttt{lpzObjectName}
Object name

Return value
Numeric value of the scale color for bar objects

See also
GetBackColor example (Page 1518)
Color chart (Page 1592)

GetSelBGColor

Function
Specifies the background color of the selected entry for the "text list" object.

Syntax
\begin{verbatim}
long int GetSelBGColor(LPCTSTR lpzPictureName, LPCTSTR lpzObjectName);
\end{verbatim}

Parameters

\texttt{lpzPictureName}
Picture name

\texttt{lpzObjectName}
Object name

Return value
Numeric value defining the background color of the selected entry

See also
GetBackColor example (Page 1518)
Color chart (Page 1592)
GetSelTextColor

Function
Specifies the font color of the selected entry for the “text list” object.

Syntax
long int GetSelTextColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name

Return value
Numeric value defining the font color of the selected entry

See also
- GetBackColor example (Page 1518)
- Color chart (Page 1592)

GetTrendColor

Function
Specifies the trend color of bar objects.

Syntax
long int GetTrendColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
Return value

Numeric value defining the trend color of bar objects

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetUnselBGColor

Function

Specifies the background color of the non-selected entries for the "text list" object.

Syntax

long int GetUnselBGColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the background color of the non-selected entries

See also

Color chart (Page 1592)
GetBackColor example (Page 1518)

GetUnselTextColor

Function

Specifies the font color of the non-selected entries for the "text list" object.

Syntax

long int GetUnselTextColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Numeric value defining the font color of the non-selected entries

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

Fill - short description

The functions in the Fill group control the dynamic filling of objects.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetFilling

Function

Specifies whether dynamic filling with background color is activated.

Syntax

```c
BOOL GetFilling(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name


**IpszObjectName**
Object name

**Return value**

**TRUE**
Dynamic filling with background color is activated.

**FALSE**
Dynamic filling with background color is not activated.

**See also**
GetFilling example (Page 1519)

**GetFillingIndex**

**Function**
Specifies the current fill level.

**Syntax**

```c
long int GetFillingIndex(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **IpszPictureName**
  Picture name
- **IpszObjectName**
  Object name

**Return value**
Current fill level as a numeric value (0 - 100)

**See also**
GetFillingIndex example (Page 1520)
Flash - short description

The various flashing properties can be modified or called in using the functions in the Flash group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetBackFlashColorOff

Function

Specifies the background flash color for the deactivated status.

Syntax

`long int GetBackFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);`

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Background flash color for the deactivated status as a numeric value

See also

- Color chart (Page 1592)
- GetFlashBackColorOn example (Page 1521)
GetBackFlashColorOn

Function

Specifies the background flash color for the activated status.

Syntax

```c
long int GetBackFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value

Background flash color for the activated status as a numeric value

See also

- GetFlashBackColorOn example (Page 1521)
- Color chart (Page 1592)

GetBorderFlashColorOff

Function

Specifies the border or line flashing color for the deactivated status.

Syntax

```c
long int GetBorderFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
Return value
Border or line flashing color for the deactivated status as a numeric value

See also
GetFlashBackColorOn example (Page 1521)
Color chart (Page 1592)

GetBorderFlashColorOn

Function
Specifies the border or line flashing color for the activated status.

Syntax
long int GetBorderFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name
lpszObjectName
Object name

Return value
Border or line flashing color for the activated status as a numeric value

See also
GetFlashBackColorOn example (Page 1521)
Color chart (Page 1592)

GetFlashBackColor

Function
Specifies whether flashing of the background is activated or not.

Syntax
BOOL GetFlashBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

_lpszPictureName_
Picture name

_lpszObjectName_
Object name

Return value

TRUE
Flashing background is activated.

FALSE
Flashing background is not activated.

See also
GetFlashBackColor example (Page 1521)

GetFlashBorderColor

Function
Specifies whether flashing of the border or line is activated or not.

Syntax
BOOL GetFlashBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_lpszPictureName_
Picture name

_lpszObjectName_
Object name

Return value

TRUE
Flashing of the border or line is activated.
FALSE
Flashing of the border or line is not activated.

See also
GetFlashBackColor example (Page 1521)

GetFlashForeColor

Function
Specifies whether flashing of the font is activated or not.

Syntax
BOOL GetFlashForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Flashing of the font is activated.

FALSE
Flashing of the font is not activated.

See also
GetFlashBackColor example (Page 1521)

GetFlashRateBackColor

Function
Specifies the flash frequency of the background.
long int GetFlashRateBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Flash frequency of the background

See also
GetFlashBackColorOn example (Page 1521)
Flash frequencies (Page 1590)

GetFlashRateBorderColor

Function
Specifies the flash frequency of the line or border.

Syntax
long int GetFlashRateBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Flash frequency of the line or border
See also

GetFlashBackColorOn example (Page 1521)
Flash frequencies (Page 1590)

GetFlashRateForeColor

Function

Specifies the flash frequency of the font.

Syntax

long int GetFlashRateForeColor(LPCSTR lpszPictureName, LPCSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Flash frequency of the font

See also

GetFlashBackColorOn example (Page 1521)
Flash frequencies (Page 1590)

GetForeFlashColorOff

Function

Specifies the font flash color for the deactivated status.

Syntax

long int GetForeFlashColorOff(LPCSTR lpszPictureName, LPCSTR lpszObjectName);
Parameters

<table>
<thead>
<tr>
<th>lpszPictureName</th>
<th>Picture name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszObjectName</td>
<td>Object name</td>
</tr>
</tbody>
</table>

Return value

Font flash color for the deactivated status as a numeric value

See also

GetFlashBackColorOn example (Page 1521)
Color chart (Page 1592)

GetForeFlashColorOn

Function

Specifies the font flash color for the activated status.

Syntax

```c
long int GetForeFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

<table>
<thead>
<tr>
<th>lpszPictureName</th>
<th>Picture name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszObjectName</td>
<td>Object name</td>
</tr>
</tbody>
</table>

Return value

Font flash color for the activated status as a numeric value

See also

GetFlashBackColorOn example (Page 1521)
Color chart (Page 1592)
focus

Focus - short description
Using the functions in the Focus group, it is possible to set the focus or poll which object has the focus.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

Get_Focus

Function
Specifies the name of the object currently or last focussed.

Syntax
char *Get_Focus();

Return value
Name of the object currently or last focussed.
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
pszValue = GetText(lpszPictureName, "Text1");
if(pszValue != NULL)
{
    ........
}

See also
GetFocus example (Page 1523)
The various properties affecting text can be modified or called in using the functions in the Font group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetAlignmentLeft

Function

Specifies the horizontal text alignment (left, centered, right).

Syntax

long int GetAlignmentLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Horizontal text alignment as a numeric value

See also

Text alignment (Page 1597)
GetFontSize example (Page 1524)
GetAlignmentTop

Function

Specifies the vertical text alignment (top, centered, bottom).

Syntax

long int GetAlignmentTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Vertical text alignment as a numeric value

See also

GetFontSize example (Page 1524)
Text alignment (Page 1597)

GetFontBold

Function

Specifies whether the font is bold or not.

Syntax

BOOL GetFontBold(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

TRUE
Bold font on

FALSE
Bold font off

See also
GetFontBold example (Page 1523)

GetFontItalic

Function
Specifies whether the font is italic or not.

Syntax
BOOL GetFontItalic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Italic font on

FALSE
Italic font off

See also
GetFontBold example (Page 1523)
GetFontName

Function

Indicates the current font name.

Syntax

char* GetFontName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Pointer to the name of the font currently selected.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}
```

See also

GetText example (Page 1555)

GetFontSize

Function

Specifies the font size.

Syntax

long int GetFontSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
GetFontUnderline

Function
Specifies whether the font is underlined or not.

Syntax
BOOL GetFontUnderline(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpzPictureName
Picture name

lpzObjectName
Object name

Return value
TRUE
Underlined font on

FALSE
Underlined font off

See also
GetFontBold example (Page 153)
GetOrientation

Function

Specifies the text orientation (vertical/horizontal).

Syntax

BOOL GetOrientation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Vertical text orientation

FALSE
Horizontal text orientation

See also

GetFontBold example (Page 1523)

GetText

Function

Specifies the value of the "text" property for objects like static text, check box or radio box.

Syntax

char* GetText(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name
**lpszObjectName**
Object name

**Return value**
Pointer to a text.
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
```c
pszValue = GetText(lpszPictureName, "Text1");
if(pszValue != NULL)
{
    .......
}
```

**Note**
In case of check and radio boxes, the element to be determined must be defined with the "SetIndex" function before actually activating this function.

**See also**
GetText example (Page 1555)

general

**GetLayer**

**Function**
Specifies the picture layer in which the object is located.

**Syntax**
```c
long int GetLayer(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

```
lpszPictureName
Picture name

lpszObjectName
Object name
```
Return value

Picture layer in which the object is located

geometry

Geometry - short description

The size, position and other geometrical properties of objects can be modified or called in using the functions in the Geometry group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetActualPointLeft

Function

Specifies the X value of the current position in a polygon or polygon line.

Syntax

```c
long int GetActualPointLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value

X value for the current point of a polygon or polygon line

Note

The current point of the polygon can be set using the SetIndex function.
See also

GetLeft example (Page 1526)

GetActualPointTop

Function

Specifies the Y value of the current position in a polygon or polygon line.

Syntax

long int GetActualPointTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Y value for the current point of a polygon or polygon line

Note

The current point of the polygon can be set using the SetIndex function.

See also

GetTop example (Page 1555)

GetBoxCount

Function

Specifies the number of fields for check boxes and radio boxes.

Syntax

long int GetBoxCount(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Number of fields in a check box or radio box.

**GetDirection**

**Function**

Specifies the bar direction for bar objects.

**Syntax**

```c
long int GetDirection(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Bar direction of bar objects as numeric value

**See also**

Bar direction (Page 1590)

**GetEndAngle**

**Function**

Specifies the end angle of circle and ellipse segments and circle and elliptical arcs.
Syntax

long int GetEndAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

End angle of circle and ellipse segments as well as circle and ellipse arcs

GetGrid

Function

Specifies whether the grid is activated in the graphics area of Graphics Designer.

Syntax

BOOL GetGrid(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Grid in Graphics Designer is activated.

FALSE
Grid in Graphics Designer is deactivated.
GetGridHeight

Function

Specifies the height of the grid in the graphics area of Graphics Designer.

Syntax

long int GetGridHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Height of the grid in Graphics Designer

GetGridWidth

Function

Specifies the width of the grid in the graphics area of Graphics Designer.

Syntax

long int GetGridWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Width of the grid in Graphics Designer
GetHeight

Function

Specifies the height of the rectangle framing an object.

Syntax

long int GetHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Height of the rectangle framing an object

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

GetHeight example (Page 1524)

GetLeft

Function

Specifies the X position of the upper left corner of the rectangle framing an object.

Syntax

long int GetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

Return value

Current X value of the upper left corner of the rectangle framing an object

See also

GetLeft example (Page 1526)

GetPointCount

Function

Specifies the number of corners of a polygon or in a polygon line.

Syntax

```c
long int GetPointCount(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

Return value

Number of corners of a polygon or in a polyline

GetRadius

Function

Specifies the radius of a circle, circle segment or arc.
**Syntax**

```c
long int GetRadius(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

Radius of a circle, circle segment or arc

**See also**

GetHeight example (Page 1524)

---

**GetRadiusHeight**

**Function**

Specifies the radius of an ellipse, ellipse segment or elliptical arc in a vertical direction.

**Syntax**

```c
long int GetRadiusHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

Radius of an ellipse, ellipse segment or elliptical arc in a vertical direction

**See also**

GetHeight example (Page 1524)
GetRadiusWidth

Function

Specifies the radius of an ellipse, ellipse segment or elliptical arc in a horizontal direction.

Syntax

```c
long int GetRadiusWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName` Picture name
- `lpszObjectName` Object name

Return value

Radius of an ellipse, ellipse segment or elliptical arc in a horizontal direction

See also

GetHeight example (Page 1524)

GetReferenceRotationLeft

Function

Specifies the X value of the rotation reference (central axis about which the object can be rotated) for lines, polygons and polylines.

Syntax

```c
long int GetReferenceRotationLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName` Picture name
- `lpszObjectName` Object name
Return value

X value of the rotation reference for lines, polygons and polygon lines

See also

GetLeft example (Page 1526)

GetReferenceRotationTop

Function

Specifies the Y value of the rotation reference (central axis about which the object can be rotated) for lines, polygons and polylines.

Syntax

```c
long int GetReferenceRotationTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

Return value

Y value of the rotation reference for lines, polygons and polygon lines

See also

GetTop example (Page 1555)

GetRotationAngle

Function

Specifies the angle of rotation about the central axis for lines, polygons and polylines.

Syntax

```c
long int GetRotationAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```
Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

Return value
Angle of rotation about the central axis

See also
GetHeight example (Page 1524)

GetRoundCornerHeight

Function
Specifies the radius of the rounded corner of a rectangle vertically.

Syntax
long int GetRoundCornerHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

Return value
Vertical radius of the rounded corner of a rectangle

See also
GetHeight example (Page 1524)
GetRoundCornerWidth

Function

Specifies the radius of the rounded corner of a rectangle horizontally.

Syntax

long int GetRoundCornerWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Horizontal radius of the corner of the rounded corner of a rectangle

See also

GetWidth example (Page 1557)

GetStartAngle

Function

Specifies the start angle of circle and ellipse segments and circle and elliptical arcs.

Syntax

long int GetStartAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value
Start angle of circle and ellipse segments as well as circle and elliptical arcs

See also
GetHeight example (Page 1524)

GetTop
Function
Specifies the Y position of the upper left corner of the rectangle framing an object.

Syntax
long int GetTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name
lpszObjectName
Object name

Return value
Current Y value of the upper left corner of the rectangle framing an object

See also
GetTop example (Page 1555)

GetWidth
Function
Specifies the width of the rectangle framing an object.

Syntax
long intGetWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Width of the rectangle framing an object

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also
GetWidth example (Page 1557)

GetZeroPoint

Function
When using bar objects, it indicates the zero point.

Syntax

long int GetZeroPoint(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Zero point for bar objects
See also

GetHeight example (Page 1524)

i_o

i_o - short description

The various properties affecting input and output values can be modified or called in using the functions in the i_o group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetAssignments

Function

Assignment of text to the value range of lists

Syntax

char* GetAssignments(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

The assignment of text to the value range depends on the list type. The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}
```
See also

List types (Page 1596)

GetAssumeOnExit

Function

Specifies for I/O fields whether the entered value is assumed upon exiting the field.

Syntax

BOOL GetAssumeOnExit(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Value application upon exiting the field.

FALSE
No value application upon exiting the field.

GetAssumeOnFull

Function

Specifies for I/O fields whether the entered value is assumed on completion of input.

Syntax

BOOL GetAssumeOnFull(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name
 IPSZObjectName
 Object name

Return value

TRUE
Value application on completion of input.

FALSE
No value application on completion of input.

GetBitNumber

Function
Specifies the relevant bit in the output value for the "bit" list type.

Syntax
long int GetBitNumber(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Indication of the relevant bit in the output value for the "bit" list type

See also
GetHiddenInput example (Page 1525)
List types (Page 1596)

GetClearOnError

Function
Specifies for I/O fields whether deletion of the content in case of input errors is activated.
Syntax

BOOL GetClearOnError(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Deletion of the content in case of input errors is activated

FALSE
Deletion of the content in case of input errors is not activated

GetClearOnNew

Function

Specifies for I/O fields whether deletion of the content on new input is activated.

Syntax

BOOL GetClearOnNew(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Deletion of the content on new input is activated.
Deletion of the content on new input is not activated.

GetDataFormat

Function
Specifies the data type of the field content for I/O fields.

Syntax
long int GetDataFormat(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName: Picture name
- lpszObjectName: Object name

Return value
Data type of the field content as numeric value

See also
GetHiddenInput example (Page 1525)
I/O field, data type of the field content (Page 1592)

GetHiddenInput

Function
Specifies whether hidden input is activated for I/O fields.

Syntax
BOOL GetHiddenInput(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

\texttt{lpszPictureName}
Picture name

\texttt{lpszObjectName}
Object name

Return value

\texttt{TRUE}
Hidden input is activated

\texttt{FALSE}
Hidden input is not activated

See also

GetHiddenInput example (Page 1525)

\textbf{GetInputValueChar}

Function

Specifies the input value in the data type "char" for I/O fields.

Syntax

\begin{verbatim}
char* GetInputValueChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
\end{verbatim}

Parameters

\texttt{lpszPictureName}
Picture name

\texttt{lpszObjectName}
Object name

Return value

Pointer to the input value in the data type "char".

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

\begin{verbatim}
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
\end{verbatim}
GetInputValueDouble

Function

Specifies the input value in the data type "double" for I/O fields.

Syntax

double GetInputValueDouble(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Input value in the data type "double"

GetListType

Function

Specifies the list type for the "text list" object.

Syntax

long int GetListType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

List type for the "text list" object

See also

GetHiddenInput example (Page 1525)
List types (Page 1596)

GetNumberLines

Function

Specifies the number of visible lines for the "text list" object.

Syntax

long int GetNumberLines(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Number of visible lines for the "text list" object

Note

If the amount of configured text is larger than the number of visible lines, the "text list" object receives a vertical scroll bar.

See also

GetHiddenInput example (Page 1525)

GetOutputFormat

Function

Specifies the output format for I/O fields.
Syntax

char* GetOutputFormat(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Pointer to the output format.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}

See also

I/O field, data type of the field content (Page 1592)
I/O field, output format (Page 1591)

GetOutputValueChar

Function

Determines the output value in the data type "char" for I/O fields. This function should only be used if the field content of the I/O field is of the "string" data type.

Syntax

char* GetOutputValueChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name
GetOutputValueDouble

Function
Determines the output value in the data type "double" for I/O fields. This function should only be used if the field content of the I/O field is not of the "string" data type.

Syntax
```c
double GetOutputValueDouble(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value
Output value in the data type "double"

See also
- GetOutputValueDouble example (Page 1529)
Limits

Limits - short description
The various properties affecting limit values can be modified or called in using the functions in the Limits group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetAlarmHigh

Function
Specifies the upper alarm limit for bar objects.

Syntax
double GetAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- `lpszPictureName` Picture name
- `lpszObjectName` Object name

Return value
Upper alarm limit for bar objects

See also
GetAlarmHigh example (Page 1518)

GetAlarmLow

Function
Specifies the lower alarm limit for bar objects.
Syntax

double GetAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Lower alarm limit for bar objects

See also

GetAlarmHigh example (Page 1518)

GetCheckAlarmHigh

Function

When using bar objects, it specifies whether the upper alarm limit is monitored.

Syntax

BOOL GetCheckAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the upper alarm limit is monitored.

FALSE
In case of bar objects the upper alarm limit is not monitored.
See also

GetMarker example (Page 1528)

GetCheckAlarmLow

Function

When using bar objects, it specifies whether the lower alarm limit is monitored.

Syntax

BOOL GetCheckAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the lower alarm limit is monitored.

FALSE
In case of bar objects the lower alarm limit is not monitored.

See also

GetMarker example (Page 1528)

GetCheckLimitHigh4

Function

When using bar objects, it specifies whether the upper limit value reserve 4 is monitored.

Syntax

BOOL GetCheckLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

`lpszPictureName`
Picture name

`lpszObjectName`
Object name

Return value

TRUE
In case of bar objects the upper limit value reserve 4 is monitored.

FALSE
In case of bar objects the upper limit value reserve 4 is not monitored.

See also

GetMarker example (Page 1528)

GetCheckLimitHigh5

Function

When using bar objects, it specifies whether the upper limit value reserve 5 is monitored.

Syntax

`BOOL GetCheckLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);`

Parameters

`lpszPictureName`
Picture name

`lpszObjectName`
Object name

Return value

TRUE
In case of bar objects the upper limit value reserve 5 is monitored.
FALSE
In case of bar objects the upper limit value reserve 5 is not monitored.

See also
GetMarker example (Page 1528)

GetCheckLimitLow4

Function
When using bar objects, it specifies whether the lower limit value reserve 4 is monitored.

Syntax
BOOL GetCheckLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

  lpszPictureName
  Picture name

  lpszObjectName
  Object name

Return value

  TRUE
  In case of bar objects the lower limit value reserve 4 is monitored.

  FALSE
  In case of bar objects the lower limit value reserve 4 is not monitored.

See also
GetMarker example (Page 1528)

GetCheckLimitLow5

Function
When using bar objects, it specifies whether the lower limit value reserve 5 is monitored.
Syntax

BOOL GetCheckLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the lower limit value reserve 5 is monitored.

FALSE
In case of bar objects the lower limit value reserve 5 is not monitored.

See also

GetMarker example (Page 1528)

GetCheckToleranceHigh

Function

When using bar objects, it specifies whether the upper tolerance limit is monitored.

Syntax

BOOL GetCheckToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

**TRUE**
In case of bar objects the upper tolerance limit is monitored.

**FALSE**
In case of bar objects the upper tolerance limit is not monitored.

See also
GetMarker example (Page 1528)

GetCheckToleranceLow

**Function**
When using bar objects, it specifies whether the lower tolerance limit is monitored.

**Syntax**

```c
BOOL GetCheckToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

**TRUE**
In case of bar objects the lower tolerance limit is monitored.

**FALSE**
In case of bar objects the lower tolerance limit is not monitored.

See also
GetMarker example (Page 1528)
GetCheckWarningHigh

Function
When using bar objects, it specifies whether the upper warning limit is monitored.

Syntax
BOOL GetCheckWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- `lpszPictureName` Picture name
- `lpszObjectName` Object name

Return value
- **TRUE**
  In case of bar objects the upper warning limit is monitored.
- **FALSE**
  In case of bar objects the upper warning limit is not monitored.

See also
GetMarker example (Page 1528)

GetCheckWarningLow

Function
When using bar objects, it specifies whether the lower warning limit is monitored.

Syntax
BOOL GetCheckWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- `lpszPictureName` Picture name
IColorAlarmHigh

Function
Specifies the bar color for bar objects upon reaching the upper alarm limit.

Syntax
long int GetColorAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Numeric value defining the bar color upon reaching the upper alarm limit

See also
GetBackColor example (Page 1518)
Color chart (Page 1592)
**GetColorAlarmLow**

**Function**
Specifies the bar color for bar objects upon reaching the lower alarm limit.

**Syntax**
```c
long int GetColorAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

**Return value**
Numeric value defining the bar color upon reaching the lower alarm limit

**See also**
- GetBackColor example (Page 1518)
- Color chart (Page 1592)

**GetColorLimitHigh4**

**Function**
Specifies the bar color for bar objects upon reaching the upper limit reserve 4.

**Syntax**
```c
long int GetColorLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
Return value

Numeric value defining the bar color upon reaching the upper limit reserve 4

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetColorLimitHigh5

Function

Specifies the bar color for bar objects upon reaching the upper limit reserve 5.

Syntax

long int GetColorLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the bar color upon reaching the upper limit reserve 5

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetColorLimitLow4

Function

Specifies the bar color for bar objects upon reaching the lower limit reserve 4.

Syntax

long int GetColorLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

\texttt{lpszPictureName}  
Picture name

\texttt{lpszObjectName}  
Object name

Return value

Numeric value defining the bar color upon reaching the lower limit reserve 4

See also

GetBackColor example (Page 1518)  
Color chart (Page 1592)

GetColorLimitLow5

Function

Specifies the bar color for bar objects upon reaching the lower limit reserve 5.

Syntax

\texttt{long int GetColorLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);}  

Parameters

\texttt{lpszPictureName}  
Picture name

\texttt{lpszObjectName}  
Object name

Return value

Numeric value defining the bar color upon reaching the lower limit reserve 5

See also

GetBackColor example (Page 1518)  
Color chart (Page 1592)
**GetColorToleranceHigh**

**Function**

Specifies the bar color for bar objects upon reaching the upper tolerance limit.

**Syntax**

```c
long int GetColorToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

**Return value**

Numeric value defining the bar color upon reaching the upper tolerance limit

**See also**

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

**GetColorToleranceLow**

**Function**

Specifies the bar color for bar objects upon reaching the lower tolerance limit.

**Syntax**

```c
long int GetColorToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
Return value

Numeric value defining the bar color upon reaching the lower tolerance limit

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetColorWarningHigh

Function

Specifies the bar color for bar objects upon reaching the upper warning limit limit.

Syntax

long int GetColorWarningHigh(LPCSTR lpszPictureName, LPCSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the bar color upon reaching the upper warning limit

See also

GetBackColor example (Page 1518)
Color chart (Page 1592)

GetColorWarningLow

Function

Specifies the bar color for bar objects upon reaching the lower warning limit.

Syntax

long int GetColorWarningLow(LPCSTR lpszPictureName, LPCSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Numeric value defining the bar color upon reaching the lower warning limit

See also

- GetBackColor example (Page 1518)
- Color chart (Page 1592)

**GetLimitHigh4**

Function

Specifies the upper limit value for reserve 4 for bar objects.

Syntax

```c
double GetLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

High limit value for reserve 4 for bar objects

See also

- GetAlarmHigh example (Page 1518)
GetLimitHigh5

Function

Specifies the upper limit value for reserve 5 for bar objects.

Syntax

double GetLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

High limit value for reserve 5 for bar objects

See also

GetAlarmHigh example (Page 1518)

GetLimitLow4

Function

Specifies the low limit value for reserve 4 for bar objects.

Syntax

double GetLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

Low limit value for reserve 4 for bar objects

See also

GetAlarmHigh example (Page 1518)

GetLimitLow5

Function

Specifies the low limit value for reserve 5 for bar objects.

Syntax

double GetLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

Return value

Low limit value for reserve 5 for bar objects

See also

GetAlarmHigh example (Page 1518)

GetLimitMax

Function

Specifies the upper limit value for I/O fields.

Syntax

double GetLimitMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

High limit value for I/O fields

See also

GetAlarmHigh example (Page 1518)

GetLimitMin

Function

Specifies the low limit value for I/O fields.

Syntax

```
double GetLimitMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Low limit value for I/O fields

See also

GetAlarmHigh example (Page 1518)
GetMarker

Function
When using bar objects, it specifies whether the limit marker is displayed.

Syntax
BOOL GetMarker(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name

Return value
- TRUE
  Limit marker for bar objects is displayed.
- FALSE
  Limit marker for bar objects is not displayed.

See also
GetMarker example (Page 1528)

GetToleranceHigh

Function
Specifies the upper tolerance limit for bar objects.

Syntax
double GetToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
GetToleranceLow

Function
Specifies the lower tolerance limit for bar objects.

Syntax
```c
double GetToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value
Lower tolerance limit for bar objects

See also
- GetAlarmHigh example (Page 1518)

GetTypeAlarmHigh

Function
Specifies for bar objects whether the upper alarm limit is given in percentages or absolute terms.
Syntax

BOOL GetTypeAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the upper alarm limit is given in percentages.

FALSE
In case of bar objects the upper alarm limit is given in absolute terms.

See also

GetMarker example (Page 1528)

GetTypeAlarmLow

Function

Specifies for bar objects whether the lower alarm limit is given in percentages or absolute terms.

Syntax

BOOL GetTypeAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

TRUE
In case of bar objects the lower alarm limit is given in percentages.

FALSE
In case of bar objects the lower alarm limit is given in absolute terms.

See also
GetMarker example (Page 1528)

GetTypeLimitHigh4

Function
Specifies for bar objects whether the upper limit reserve 4 is given in percentages or absolute terms.

Syntax
BOOL GetTypeLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the upper limit reserve 4 is given in percentages.

FALSE
In case of bar objects the upper limit reserve 4 is given in absolute terms.

See also
GetMarker example (Page 1528)
GetTypeLimitHigh5

Function
Specifies for bar objects whether the upper limit reserve 5 is given in percentages or absolute terms.

Syntax
BOOL GetTypeLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name

Return value
- TRUE
  In case of bar objects the upper limit reserve 5 is given in percentages.
- FALSE
  In case of bar objects the upper limit reserve 5 is given in absolute terms.

See also
GetMarker example (Page 1528)

GetTypeLimitLow4

Function
Specifies for bar objects whether the lower limit reserve 4 is given in percentages or absolute terms.

Syntax
BOOL GetTypeLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

- **TRUE**
  In case of bar objects the lower limit reserve 4 is given in percentages.

- **FALSE**
  In case of bar objects the lower limit reserve 4 is given in absolute terms.

**See also**

GetMarker example (Page 1528)

---

**GetTypeLimitLow5**

**Function**

Specifies for bar objects whether the lower limit reserve 5 is given in percentages or absolute terms.

**Syntax**

```c
BOOL GetTypeLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**

- **TRUE**
  In case of bar objects the lower limit reserve 5 is given in percentages.
FALSE
In case of bar objects the lower limit reserve 5 is given in absolute terms.

See also
GetMarker example (Page 1528)

GetTypeToleranceHigh

Function
Specifies for bar objects whether the upper tolerance limit is given in percentages or absolute terms.

Syntax
BOOL GetTypeToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

Return value
TRUE
In case of bar objects the upper tolerance limit is given in percentages.

FALSE
In case of bar objects the upper tolerance limit is given in absolute terms.

See also
GetMarker example (Page 1528)

GetTypeToleranceLow

Function
Specifies for bar objects whether the lower tolerance limit is given in percentages or absolute terms.
Syntax

BOOL GetTypeToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
In case of bar objects the lower tolerance limit is given in percentages.

FALSE
In case of bar objects the lower tolerance limit is given in absolute terms.

See also

GetMarker example (Page 1528)

GetTypeWarningHigh

Function

Specifies for bar objects whether the upper warning limit is given in percentages or absolute terms.

Syntax

BOOL GetTypeWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

**TRUE**
In case of bar objects the upper warning limit is given in percentages.

**FALSE**
In case of bar objects the upper warning limit is given in absolute terms.

See also

GetMarker example (Page 1528)

**GetTypeWarningLow**

Function

Specifies for bar objects whether the lower warning limit is given in percentages or absolute terms.

Syntax

```c
BOOL GetTypeWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`
  Picture name
- `lpszObjectName`
  Object name

Return value

**TRUE**
In case of bar objects the lower warning limit is given in percentages.

**FALSE**
In case of bar objects the lower warning limit is given in absolute terms.

See also

GetMarker example (Page 1528)
GetWarningHigh

Function

Specifies the upper warning limit for bar objects.

Syntax

double GetWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Upper warning limit for bar objects

See also

GetAlarmHigh example (Page 1518)

GetWarningLow

Function

Specifies the lower warning limit for bar objects.

Syntax

double GetWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

Lower warning limit for bar objects

See also

GetAlarmHigh example (Page 1518)

link

Link - short description

A tag link property can be created or called in using the functions in the Link group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetLink

Function

Specifies the current tag connection of object properties.

Syntax

BOOL GetLink(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, LPLINKINFO *pLink);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lpszPropertyName
Object property

pLink
Pointer to a structure of the type: LINKINFO
Return value

TRUE  
The function has been completed without any errors.

FALSE  
An error has occurred.

See also

Structure definition LINKINFO (Page 1605)
GetLink example (Page 1526)

miscs

Miscs - short description

The properties of objects can be modified or called in using the functions in the Miscs group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetAdaptBorder

Function

Specifies for static texts, I/O fields, check boxes and radio boxes whether the border of the field is to be dynamically adapted to the text size.

Syntax

BOOL GetAdaptBorder(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName  
Picture name

lpszObjectName  
Object name
Return value

**TRUE**
Border is adapted

**FALSE**
Border is not adapted

See also

GetVisible example (Page 1556)

GetAdaptPicture

Function
Specifies for picture windows whether the picture is to be adapted to the window size.

Syntax

```c
BOOL GetAdaptPicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

**TRUE**
Picture is adapted

**FALSE**
Picture is not adapted

See also

GetVisible example (Page 1556)
GetAdaptSize

Function
Specifies for picture windows whether the window is to be adapted.

Syntax
BOOL GetAdaptSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name

Return value
- TRUE
  Window is adapted
- FALSE
  Window is not adapted

See also
GetVisible example (Page 1556)

GetAverage

Function
When using bar objects, it specifies whether value averaging is activated.

Syntax
BOOL GetAverage(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
lpszObjectName
Object name

Return value

TRUE
Averaging is activated for bar objects

FALSE
Averaging is not activated for bar objects

See also

GetVisible example (Page 1556)

GetBoxType

Function
Specifies the field type (input field, output field, input/output field) for I/O fields.

Syntax

long int GetBoxType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Field type of an I/O field

See also
I/O field, field type (Page 1592)
GetCaption

Function

Specifies whether a picture or application window has a title.

Syntax

BOOL GetCaption(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Picture/application window has a title

FALSE
Picture/application window has no title

See also

GetVisible example (Page 1556)

GetCloseButton

Function

When using a picture window, it specifies whether the window can be closed.

Syntax

BOOL GetCloseButton(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name
GetColorChangeType

Function

When using bar objects, it specifies whether the color change upon reaching a limit value only affects a bar segment or the entire bar.

Syntax

BOOL GetColorChangeType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Color change applies to the bar segment

FALSE
Color change applies to the entire bar
See also

GetVisible example (Page 1556)

GetCursorControl

Function

Specifies whether cursor control is activated for I/O fields.

Syntax

BOOL GetCursorControl(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
  Picture name

lpszObjectName
  Object name

Return value

TRUE
  Cursor control for I/O fields is enabled.

FALSE
  Cursor control for I/O fields is disabled.

See also

GetVisible example (Page 1556)

GetCursorMode

Function

Specifies whether the cursor mode for the picture is alpha cursor or tab order cursor.

Syntax

BOOL GetCursorMode(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpzPictureName**
  Picture name

- **lpzObjectName**
  Object name

Return value

- **TRUE**
  Cursor mode for the picture is "Alpha-cursor"

- **FALSE**
  Cursor mode for the picture is "tab order cursor"

See also

GetVisible example (Page 1556)

GetEditAtOnce

Function

Specifies whether the "Immediate input" property is activated for I/O fields.

Syntax

```c
BOOL GetEditAtOnce(LPCTSTR lpzPictureName, LPCTSTR lpzObjectName);
```

Parameters

- **lpzPictureName**
  Picture name

- **lpzObjectName**
  Object name

Return value

- **TRUE**
  "Immediate input" property is activated
FALSE
"Immediate input" property is deactivated

See also
GetVisible example (Page 1556)

GetExtendedOperation

Function
Specifies whether the "Extended operation" property is activated for slider objects.

Syntax
BOOL GetExtendedOperation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
"Extended operation" property is activated

FALSE
"Extended operation" property is deactivated

See also
GetVisible example (Page 1556)

GetHotkey

Function
Specifies the key combination for check boxes.
Syntax

```c
long int GetHotkey(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name

Return value

Key code for key combinations for check boxes

GetHysteresis

Function

When using bar objects, it specifies whether the display appears with or without hysteresis.

Syntax

```c
BOOL GetHysteresis(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name

Return value

- **TRUE**: Display with hysteresis for bar objects
- **FALSE**: Display without hysteresis for bar objects

See also

GetVisible example (Page 1556)
GetHysteresisRange

Function

Specifies the hysteresis value in the display for bar objects.

Syntax

double GetHysteresisRange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Hysteresis in the display for bar objects

GetLanguageSwitch

Function

Specifies for the "Text list" object whether the assignment texts are to be stored in the text library or in the object itself.

Syntax

BOOL GetLanguageSwitch(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

**TRUE**
Assignment texts are stored in the text library

**FALSE**
Assignment texts are stored in the text list object

See also

GetVisible example (Page 1556)

GetLastChange

Function

Specifies the date when the picture was last changed.

Syntax

```c
char* GetLastChange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Date of the last change of the picture.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}
```

See also

GetPictureName example (Page 1531)
GetMax

Function
Specifies the maximum value for bar and slider objects.

Syntax
double GetMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Maximum value for bar and slider objects.

GetMaximizeButton

Function
Specifies for picture or application windows whether the window can be maximized.

Syntax
BOOL GetMaximizeButton(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
TRUE
Picture or application window can be maximized
FALSE
Picture or application window cannot be maximized

See also
GetVisible example (Page 1556)

GetMin
Function
Specifies the minimum value for bar and slider objects.

Syntax
double GetMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name
lpszObjectName
Object name

Return value
Minimum value for bar and slider objects

GetMoveable
Function
Specifies for picture or application windows whether the window can be moved.

Syntax
BOOL GetMoveable(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name
\textbf{GetOffsetLeft}

\textbf{Function}

Specifies the horizontal picture distance from the left window border for picture windows.

\textbf{Syntax}

\begin{verbatim}
long int GetOffsetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
\end{verbatim}

\textbf{Parameters}

\begin{itemize}
  \item \textbf{lpszPictureName}
    \begin{itemize}
      \item Picture name
    \end{itemize}
  \item \textbf{lpszObjectName}
    \begin{itemize}
      \item Object name
    \end{itemize}
\end{itemize}

\textbf{Return value}

Horizontal picture distance from the left window border for picture windows

\textbf{GetOffsetTable}

\textbf{Function}

Specifies the vertical picture distance from the upper window border for picture windows.
Syntax

long int GetOffsetTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Vertical picture distance from the upper window border for picture windows

GetOnTop

Function

Specifies for picture or application windows whether the window is always in the foreground.

Syntax

BOOL GetOnTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Picture or application window is always in the foreground

FALSE
Picture or application window can be overlapped by other windows.

See also

GetVisible example (Page 1556)
GetOperation

Function

Specifies whether the object can be operated.

Syntax

BOOL GetOperation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Object is operable

FALSE
Object is not operable

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

GetVisible example (Page 1556)

GetOperationMessage

Function

Specifies for I/O fields, check boxes, radio boxes or sliders whether a message is output following operation.

Syntax

BOOL GetOperationMessage(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

- **TRUE**
  Upon operation a message is issued

- **FALSE**
  Upon operation no message is issued

See also

GetVisible example (Page 1556)

GetOperationReport

Function

Specifies for all objects except application and picture windows and OLE control whether the reason for the operation is logged.

Syntax

```c
BOOL GetOperationReport(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

- **TRUE**
  Reason for the operation is logged.
FALSE
Reason for the operation is not logged.

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also
GetVisible example (Page 1556)

GetPasswordLevel

Function
Specifies the authorization level for the operation of the object for all objects except application and picture windows and OLE control.

Syntax
long int GetPasswordLevel(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Authorization level for the operation of the object

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.
GetPictureName

Function

Returns the name of the picture currently displayed in the picture window.

Syntax

char* GetPictureName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Name of the picture window

Return value

Pointer to the name of the currently displayed picture

Note

If both parameters are NULL, a pointer appears indicating the name of the basic screen.

See also

GetPictureName example (Page 1531)

GetProcess

Function

Specifies the default setting value for the process value to be displayed for bar and slider objects.

Specifies the selected fields for check boxes and radio boxes.

Syntax

double GetProcess(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

Return value

- For bar and slider objects: Default setting value for the process value to be displayed
- For check and radio boxes: In a 32-bit word each field is represented by a bit (field 1 corresponds to the bit value 0). Selected fields are marked by a set bit. Non-existing are assigned 0.

GetScrollBars

Function

Specifies for picture windows whether the window has a scroll bar.

Syntax

BOOL GetScrollBars(LPCTSTR lpzPictureName, LPCTSTR lpzObjectName);

Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

Return value

TRUE
Picture window has a scroll bar

FALSE
Picture window has no scroll bar

See also

GetVisible example (Page 1556)
GetServerName

Function

Specifies the default setting for the process value to be displayed for OLE control and OLE object.

Syntax

```c
char* GetServerName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name

Return value

Name of the object (OLE control and OLE object) under which it is registered in WINDOWS. The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}
```

See also

GetPictureName example (Page 1531)

GetSizeable

Function

Specifies for application or picture windows whether the window size can be changed.

Syntax

```c
BOOL GetSizeable(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```
Parameters

\texttt{lpzPictureName}
Picture name

\texttt{lpzObjectName}
Object name

Return value

\texttt{TRUE}
Application or picture window is sizeable

\texttt{FALSE}
Application or picture window is not sizeable

See also

GetVisible example (Page 1556)

GetSmallChange

Function

Specifies the number of steps for slider objects by which the slider is shifted by a mouse click.

Syntax

\texttt{long\ int\ GetSmallChange(LPCTSTR \texttt{lpzPictureName}, \texttt{LPCTSTR \texttt{lpzObjectName});}

Parameters

\texttt{lpzPictureName}
Picture name

\texttt{lpzObjectName}
Object name

Return value

Number of steps by which the slider is shifted by a mouse click
GetTagPrefix

Function

Returns the tag prefix of a picture window.

Syntax

char* GetTagPrefix(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Tag prefix of the picture window.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}

See also

GetTagPrefix example (Page 1549)

GetTrend

Function

When using bar objects, it specifies whether the trend display is activated.

Syntax

BOOL GetTrend(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

\texttt{lpszPictureName}  
Picture name

\texttt{lpszObjectName}  
Object name

Return value

\textbf{TRUE}  
Trend display is activated for a bar object

\textbf{FALSE}  
Trend display is not activated for a bar object

See also

GetVisible example (Page 1556)

GetUpdateCycle

Function

Specifies the update cycle for the entire picture.

Syntax

long int GetUpdateCycle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

\texttt{lpszPictureName}  
Picture name

\texttt{lpszObjectName}  
Object name

Return value

Numeric value defining the update cycle

See also

Structure definition LINKINFO (Page 1605)
GetVisible

Function

Specifies whether the object is displayed.

Syntax

BOOL GetVisible(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Object is displayed

FALSE

Object is not displayed

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

GetVisible example (Page 1556)

GetWindowBorder

Function

Specifies for application or picture windows whether the object is displayed with a border.

Syntax

BOOL GetWindowBorder(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Application or picture window is displayed with a border.

FALSE
Application or picture window is displayed without a border.

See also

GetVisible example (Page 1556)

GetZeroPointValue

Function
Specifies the absolute value of the zero point for bar objects.

Syntax

double GetZeroPointValue(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Absolute value of the zero point for the bar display
GetZoom

Function

Specifies the scaling factor for picture windows.

Syntax

long int GetZoom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Scaling factor of a picture window

ole_control

OLE_control - short description

The functions in the ole_Control group can only be used with OCX slider objects.
Various OCX slider object properties and settings can be modified or called in using these functions.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetPosition

Function

Specifies the position of the slider for OCX slider objects.
Syntax

long int GetPosition(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Slider position of the OCX slider object as numeric value

See also

GetPosition example (Page 1532)

GetRangeMax

Function

Specifies the adjustment range "Max" for OCX slider objects.

Syntax

long int GetRangeMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Adjustment range "Max" of the OCX slider object as numeric value

See also

GetRangeMax example (Page 1534)
GetRangeMin

Function

Specifies the adjustment range "Min" for OCX slider objects.

Syntax

long int GetRangeMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Adjustment range "Min" of the OCX slider object as numeric value

See also

GetRangeMin example (Page 1535)

pictures

Pictures - short description

Various properties of pictures of graphic objects and round buttons can be modified or called in using the functions in the Pictures group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetPicDeactReferenced

Function

Specifies whether the picture for the "deactivated" status is referenced for round buttons.
Syntax
BOOL GetPicDeactReferenced(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

  lpszPictureName
  Picture name

  lpszObjectName
  Object name

Return value

  TRUE
  The picture assigned to the "deactivated" status was not stored in the object.

  FALSE
  The picture assigned to the "deactivated" status was stored in the object.

GetPicDeactTransparent

Function
Specifies the transparent color for the "deactivated" status of round buttons.

Syntax
long int GetPicDeactTransparent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

  lpszPictureName
  Picture name

  lpszObjectName
  Object name
Returns value
Numeric value defining the transparent color for the "deactivated" status

Note
This function only applies to Bitmap graphics (*.bmp).

See also
Color chart (Page 1592)
GetBackColor example (Page 1518)

GetPicDeactUseTransColor

Function
Specifies whether the transparent color for the "deactivated" status is used for round buttons.

Syntax
BOOL GetPicDeactUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name
lpszObjectName
Object name

Return value
TRUE
Transparent color for "deactivated" status is used
FALSE
Transparent color for "deactivated" status is not used

GetPicDownReferenced

Function
Specifies whether the picture for the "On/pressed" status is referenced for round buttons.
Syntax

BOOL GetPicDownReferenced(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_**lpszPictureName**_
Picture name

_**lpszObjectName**_
Object name

Return value

TRUE
The picture assigned to the "On/pressed" status was not stored in the object.

FALSE
The picture assigned to the "On/pressed" status was stored in the object.

GetPicDownTransparent

Function

Specifies the transparent color for the "On/pressed" status of round buttons.

Syntax

long int GetPicDownTransparent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_**lpszPictureName**_
Picture name

_**lpszObjectName**_
Object name
Return value
Numeric value defining the transparent color for the "On/pressed" status

Note
This function only applies to Bitmap graphics (*.bmp).

See also
Color chart (Page 1592)
GetPictureDown example (Page 1530)

GetPicDownUseTransColor

Function
Specifies whether the transparent color for the "On/pressed" status is used for round buttons.

Syntax
BOOL GetPicDownUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

Return value
TRUE
Transparent color for "On/pressed" status is used

FALSE
Transparent color for "On/pressed" status is not used

GetPicReferenced

Function
When using graphic objects, it specifies whether the picture is referenced.
Syntax

BOOL GetPicReferenced(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
The assigned picture was not stored in the object.

FALSE
The assigned picture was stored in the object.

GetPicTransColor

Function

Specifies the transparent color for a background picture for graphic objects.

Syntax

long int GetPicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name
Return value

Numeric value defining the background picture of a graphic object

Note

This function only applies to Bitmap graphics (*.bmp).

See also

Color chart (Page 1592)
GetBackColor example (Page 1518)

GetPictureDeactivated

Function

Specifies the picture name for the "deactivated" status of round buttons.

Syntax

char* GetPictureDeactivated(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName  
Picture name

lpszObjectName  
Object name

Return value

Picture name for "deactivated" status.
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}

Note

Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.
GetPictureDown

Function

Specifies the picture name for the "On/pressed" status of round buttons.

Syntax

char* GetPictureDown(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Picture name for the "On/pressed" status.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}
```

Note

Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also

GetPictureDown example (Page 1530)

GetPictureUp

Function

Specifies the picture name for the "Off/not pressed" status of round buttons.
Syntax

char* GetPictureUp(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

Return value

Picture name for the "Off/not pressed" status.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

pszValue = GetText(lpszPictureName, "Text1");
if(pszValue != NULL)
{
    .......
}

Note

Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also

GetPictureUp example (Page 1532)

GetPicUpReferenced

Function

Specifies whether the picture for the "Off/not pressed" status is referenced for round buttons.

Syntax

BOOL GetPicUpReferenced(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

_lpszPictureName
Picture name
IpzObjectName
Object name

Return value

TRUE
The picture assigned to the "Off/not pressed" status was not stored in the object.

FALSE
The picture assigned to the "Off/not pressed" status was stored in the object.

GetPicUpTransparent

Function
Specifies the transparent color for the "Off/not pressed" status of round buttons.

Syntax
long int GetPicUpTransparent(LPCTSTR IpzPictureName, LPCTSTR IpzObjectName);

Parameters

IpzPictureName
Picture name

IpzObjectName
Object name

Return value
Numeric value defining the transparent color for the "Off/not pressed" status

Note
This function only applies to Bitmap graphics (*.bmp).

See also
Color chart (Page 1592)
GetBackColor example (Page 1518)
GetPicUpUseTransColor

Function
Specifies whether the transparent color for the "Off/not pressed" status is used for round buttons.

Syntax
BOOL GetPicUpUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name

Return value
- TRUE
  Transparent color for "Off/not pressed" status is used
- FALSE
  Transparent color for "Off/not pressed" status is not used

GetPicUseTransColor

Function
When using graphic objects, it specifies whether the transparent color is used for a background picture.

Syntax
BOOL GetPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters
- lpszPictureName
  Picture name
IpszObjectName
Object name

Return value

TRUE
Transparent color is used for a background picture.

FALSE
Transparent color is not used for a background picture.

Property

Property - short description
The properties of objects for which there are no direct functions can be modified or called in using the functions in the Property group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetPropBOOL

Function
Specifies the current status of a property of the data type "BOOL".

Syntax
BOOL GetPropBOOL(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName)

Parameters

IpszPictureName
Picture name

IpszObjectName
Object name
GetPropChar

Function

Specifies the current status of a property of the data type "char".

Syntax

```c
char* GetPropChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName)
```

Parameters

- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name
- `lpszPropertyName`: Name of the object property

Return value

Pointer to a character string containing the value of the object property.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}
```
See also

GetPropChar example (Page 1534)

GetPropDouble

Function

Specifies the current status of a property of the data type "double".

Syntax

```c
double GetPropDouble(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName)
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lpszPropertyName**
  Name of the object property

Return value

Value of the attribute in the data type "double"

GetPropWord

Function

Specifies the current status of a property of the data type "long".

Syntax

```c
long GetPropWord(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName)
```

Parameters

- **lpszPictureName**
  Picture name
IpszObjectName
Object name

IpszPropertyName
Name of the object property

Return value
Value of the attribute in the type "long"

state

State - short description
The properties of status displays can be modified or called in using the functions in the State group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

GetBasePicReferenced

Function
Specifies whether the basic picture is referenced for the status display.

Syntax
BOOL GetBasePicReferenced(LPCTSTR IpszPictureName, LPCTSTR IpszObjectName);

Parameters
IpszPictureName
Picture name

IpszObjectName
Object name
Return value

**TRUE**
The basic picture was not stored in the object.

**FALSE**
The basic picture was stored in the object.

GetBasePicTransColor

**Function**
Specifies the transparent color of the basic picture for the status display.

**Syntax**
```c
long int GetBasePicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

**Return value**
Transparent color of the basic picture as numeric value

**Note**
This function only applies to Bitmap graphics (*.bmp).

**See also**
Color chart (Page 1592)
GetBackColor example (Page 1518)

GetBasePicture

**Function**
Specifies the basic picture name for the status display.
Syntax

```c
char* GetBasePicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  - Picture name
- **lpszObjectName**
  - Object name

Return value

Basic picture name for the status display.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}
```

GetBasePicUseTransColor

Function

When using the status display, it specifies whether the transparent color is used for the basic picture.

Syntax

```c
BOOL GetBasePicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  - Picture name
- **lpszObjectName**
  - Object name
Return value

TRUE
Transparent color is used for the basic picture.

FALSE
Transparent color is not used for the basic picture.

GetFlashFlashPicture

Function
Specifies whether the flash picture of the status display is animated dynamically or statically.

Syntax
BOOL GetFlashFlashPicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
The flash picture is animated dynamically.

FALSE
The flash picture is animated statically.

GetFlashPicReferenced

Function
Specifies whether the flash picture is referenced for the status display.

Syntax
BOOL GetFlashPicReferenced(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

\textbf{lpszPictureName}  
Picture name

\textbf{lpszObjectName}  
Object name

Return value

\textbf{TRUE}  
The flash picture was not stored in the object.

\textbf{FALSE}  
The flash picture was stored in the object.

\textbf{GetFlashPicTransColor}

Function

Specifies the transparent color of the flash picture for the status display.

Syntax

\texttt{long int GetFlashPicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);}  

Parameters

\textbf{lpszPictureName}  
Picture name

\textbf{lpszObjectName}  
Object name

Return value

Transparent color of the flash picture as numeric value

\textbf{Note}

This function only applies to Bitmap graphics (*.bmp).
See also

Color chart (Page 1592)
GetBackColor example (Page 1518)

GetFlashPicture

Function

Specifies the flash picture name for the status display.

Syntax

char* GetFlashPicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Flash picture name (file name of the graphic).
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
   .......
}

GetFlashPicUseTransColor

Function

When using the status display, it specifies whether the transparent color is used for the flash picture.

Syntax

BOOL GetFlashPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

- **TRUE**
  Transparent color is used for the flash picture.

- **FALSE**
  Transparent color is not used for the flash picture.

GetFlashRateFlashPic

Function

Specifies the flash frequency of the flash picture for the status display.

Syntax

```c
long int GetFlashRateFlashPic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

Flash frequency of a flash picture as numeric value

Note

Since the flashing is performed by means of software engineering, the precise frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time etc.).
See also

Flash frequencies (Page 1590)
GetFlashRateFlashPic example (Page 1522)

GetIndex

Function

Specifies the index of the current position in a polygon or polygon line.
Specifies the index of the current field for check boxes and radio boxes.

Syntax

long int GetIndex(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Index of the current point or field

Style - short description

Various properties affecting the appearance of objects can be modified or called in using the functions in the Style group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
GetBackBorderWidth

Function

Specifies the frame width of 3D frames and slider objects.

Syntax

```c
long int GetBackBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name

Return value

Numeric value for the frame width of 3D frames and slider objects

See also

GetBorderStyle example (Page 1519)

GetBorderEndStyle

Function

Specifies the type of line end.

Syntax

```c
long int GetBorderEndStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name
Return value

Type of line end as numeric value

See also

GetBorderStyle example (Page 1519)
Line end style (Page 1595)

GetBorderStyle

Function

Specifies the line or border style.

Syntax

long int GetBorderStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Numeric value defining the line or border style

See also

GetBorderStyle example (Page 1519)
Line styles (Page 1595)

GetBorderWidth

Function

Specifies the line or border line width.

Syntax

long int GetBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
GetBoxAlignment

Function
Specifies the arrangement of controls (left or right justified) in check boxes or radio boxes.

Syntax
long int GetBoxAlignment(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value
Numeric value defining the arrangement of controls in check boxes or radio boxes

See also
GetBorderStyle example (Page 1519)
Text alignment (Page 1597)
GetFillStyle

Function

Specifies the type of fill pattern.

Syntax

long int GetFillStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

Type of fill pattern as numeric value

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

Fill pattern (Page 1594)
GetFillStyle example (Page 1520)

GetFillStyle2

Function

Specifies the bar fill pattern for a bar graph.

Syntax

long int GetFillStyle2(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value

Bar fill pattern as numeric value

See also

- Fill pattern (Page 1594)
- GetFillStyle example (Page 1520)

**GetItemBorderStyle**

Function

Specifies the dividing line style for the "text list" object.

Syntax

```c
long int GetItemBorderStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name

Return value

Dividing line style for the "text list" object

See also

- GetBorderStyle example (Page 1519)
- Line styles (Page 1595)
GetItemBorderWidth

Function

Specifies the dividing line width for the "text list" object.

Syntax

long int GetItemBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name

Return value

Numeric value defining the dividing line width for the "text list" object

See also

GetBorderStyle example (Page 1519)

GetPressed

Function

Specifies for buttons or round buttons whether the switch setting is "pressed" or "not pressed".

Syntax

BOOL GetPressed(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
Return value

TRUE
Switch setting is "pressed"

FALSE
Switch setting is "not pressed"

GetToggle

Function
Specifies for buttons or round buttons whether the switch is latchable or not.

Syntax

BOOL GetToggle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Return value

TRUE
Switch is latchable

FALSE
Switch is not latchable

GetWindowsStyle

Function
Specifies whether buttons are to be displayed in Windows style.

Syntax

BOOL GetWindowsStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
Parameters

**lpzPictureName**
Picture name

**lpzObjectName**
Object name

Return value

**TRUE**
Button is displayed in the usual Windows fashion.

**FALSE**
The appearance of the button is defined by the user.

set

axes

Axes - short description

The functions in the Axes group can only be used with bar graph objects.
This function can be used to modify or query various bar graph object properties.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetAlignment

Function

When using bar objects, it indicates whether the text is to the right or left of the bar.

Syntax

```c
BOOL SetAlignment(LPCTSTR lpzPictureName, LPCTSTR lpzObjectName, BOOL bAlignment);
```
Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

bAlignment
Text alignment

    TRUE  Text is to the right of the bar
    FALSE Text is to the left of the bar

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetScaling example (Page 1569)

SetAxisSection

Function

When using bar objects, it specifies the axis section, i.e. the difference between the values of two neighboring axis labels.

Syntax

BOOL SetAxisSection(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAxisSection);

Parameters

lpzPictureName
Picture name

lpzObjectName
Object name
dAxisSection
Axis section

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetScaling example (Page 1569)

SetExponent

Function
Sets the axis label display for bar objects (exponential/decimal).

Syntax
BOOL SetExponent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bExponent);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bExponent
Axis labeling

TRUE Axis label in exponential form
FALSE Axis label in decimal form

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

See also
SetScaling example (Page 1569)

SetLeftComma

Function
When using bar objects, it specifies the number of integers in the axis label.

Syntax
BOOL SetLeftComma(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLeftComma);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lLeftComma
Number of integers

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetScaling example (Page 1569)
SetLongStrokesBold

Function
When using bar objects, it specifies whether the main division lines are bold or regular.

Syntax
BOOL SetLongStrokesBold(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bLongStrokesBold);

Parameters
- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name
- `bLongStrokesBold`: Main division lines on the bar graph scale
  - TRUE: bold
  - FALSE: normal

Return value
- TRUE: The function has been completed without any errors.
- FALSE: An error has occurred.

See also
SetScaling example (Page 1569)

SetLongStrokesOnly

Function
When using bar objects, it specifies whether intermediate or only main division lines are used on the scale.
Syntax

BOOL SetLongStrokesOnly(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bLongStrokesOnly);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bLongStrokesOnly
Only main division lines yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetScaling example (Page 1569)

SetLongStrokesSize

Function

When using bar objects, it specifies the length of the main division lines on the bar graph scale.

Syntax

BOOL SetLongStrokesSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLongStrokesSize);

Parameters

lpszPictureName
Picture name
IpszObjectName
Object name

ILongStrokesSize
Length of the main division marks in pixels

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetScaling example (Page 1569)

SetRightComma

Function
When using bar objects, it specifies the number of decimal places in the axis label.

Syntax
BOOL SetRightComma(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRightComma);

Parameters

IpszPictureName
Picture name

IpszObjectName
Object name

IRightComma
Number of decimal places

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

See also
SetScaling example (Page 1569)

SetScaleTicks

Function
When using bar objects, it specifies the scale marks as number of scale sections. A scale section is a part of the scale bounded by two main tick marks.

Syntax
BOOL SetScaleTicks(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScaleTicks);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

lScaleTicks
Number of scale sections

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the number of scale sections is given as 0, the bar object itself calculates a suitable scale unit.
See also

SetScaling example (Page 1569)

SetScaling

Function

Switches the bar graph scale of bar objects on or off.

Syntax

BOOL SetScaling(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bScaling);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bScaling
Scale on/off.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetScaling example (Page 1569)

SetScalingType

Function

When using bar objects, it specifies the type of bar scaling.
Syntax

BOOL SetScalingType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScalingType);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lScalingType
Type of bar scaling as numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Bar Scaling (Page 1590)
SetScaling example (Page 1569)

color

Color - short description

The various color properties of objects can be modified or queried using the functions in the Color group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
SetBackColor

Function

Sets the background color of the object.

Syntax

BOOL SetBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lBackColor
Background color of the object as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
SetBackColor2

Function
Sets the bar color for bar objects.

Syntax
BOOL SetBackColor2(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor2);

Parameters
- lpszPictureName: Picture name
- lpszObjectName: Object name
- lBackColor2: Numeric value defining the bar color

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

See also
- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetBackColor3

Function
Sets the bar background color for bar objects.

Syntax
BOOL SetBackColor3(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor3);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lBackColor3**
  Numeric value defining the bar background color

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetBackColorBottom

Function

Sets the background color of the slider objects at the bottom right.

Syntax

```c
BOOL SetBackColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColorBottom);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lBackColorBottom**
  Numeric value defining the background color of slider objects
**SetBackColorTop**

**Function**
Sets the background color of the slider objects at the top left.

**Syntax**

```
BOOL SetBackColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColorTop);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lBackColorTop**
  Numeric value defining the background color of slider objects

**Return value**

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.
See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetBorderBackColor

Function

Sets the background color of the lines or borders.

Syntax

```c
BOOL SetBorderBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderBackColor);
```

Parameters

- `lpszPictureName` Picture name
- `lpszObjectName` Object name
- `lBorderBackColor` Background color of the lines or borders

Return value

- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
SetBorderColor

Function
Sets the color of the lines or borders.

Syntax
BOOL SetBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColor);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lBorderColor**
  Numeric value defining the color of lines or borders

Return value
- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

See also
- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetBorderColorBottom

Function
Sets the 3D border color at the bottom.

Syntax
BOOL SetBorderColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColorBottom);
Parameters

\texttt{lpszPictureName}
Picture name

\texttt{lpszObjectName}
Object name

\texttt{lBorderColorBottom}
Numeric value defining the 3-D border color at the bottom

Return value

\texttt{TRUE}
The function has been completed without any errors.

\texttt{FALSE}
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetBorderColorTop

Function
Sets the 3D border color at the top.

Syntax

\texttt{BOOL SetBorderColorTop(LPCSTR \ lpszPictureName, LPCSTR \ lpszObjectName, long int \ lBorderColorTop);}  

Parameters

\texttt{lpszPictureName}
Picture name

\texttt{lpszObjectName}
Object name

\texttt{lBorderColorTop}
Numeric value defining the 3-D border color at the top
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetButtonColor

Function

Sets the button color of slider objects.

Syntax

BOOL SetButtonColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lButtonColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lButtonColor
Numeric value defining the button color of slider objects

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetColorBottom

Function

When using slider objects, it sets the color of the bottom limit.

Syntax

BOOL SetColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long intIColorBottom);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

IColorBottom
Numeric value defining the color of the bottom limit of slider objects

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
SetColorTop

Function
When using slider objects, it sets the color of the top limit.

Syntax
BOOL SetColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorTop);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name
- lColorTop
  Numeric value defining the color of the top limit of slider objects

Return value
- TRUE
  The function has been completed without any errors.
- FALSE
  An error has occurred.

See also
- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetFillColor

Function
Sets the color of the fill pattern.

Syntax
BOOL SetFillColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillColor);

See also
- Color chart (Page 1592)
- SetBackColor example (Page 1559)
Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**lFillColor**
Numeric value of the fill color

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

**Note**
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetForeColor

**Function**
Sets the font color.

**Syntax**

```c
BOOL SetForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeColor);
```

**Parameters**

**lpszPictureName**
Picture name
SetItemBorderBackColor

Function

Sets the background color of the separating line for the "text list" object.

Syntax

BOOL SetItemBorderBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderBackColor);

Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**lItemBorderBackColor**
Background color of the dividing line as a numeric value
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetItemBorderColor

Function
Sets the color of the dividing line for the "text list" object.

Syntax

BOOL SetItemBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lItemBorderColor
Numeric value defining the dividing line color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetScaleColor

Function

Sets the scale color for bar objects.

Syntax

BOOL SetScaleColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScaleColor);

Parameters

- **lpszPictureName**
  - Picture name

- **lpszObjectName**
  - Object name

- **lScaleColor**
  - Numeric value of the scale color for bar objects

Return value

- **TRUE**
  - The function has been completed without any errors.

- **FALSE**
  - An error has occurred.

See also

- Color chart (Page 1592)
- SetBackColor example (Page 1559)
SetSelBGColor

Function

Sets the background color of the selected entry for the "text list" object.

Syntax

BOOL SetSelBGColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSelBGColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lSelBGColor
Numeric value defining the background color in the selected entry

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetSelTextColor

Function

Sets the font color of a selected entry for the "text list" object.

Syntax

BOOL SetSelTextColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSelTextColor);
Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

- `lSelTextColor`
  Numeric value defining the font color in the selected entry

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetTrendColor

Function

Sets the trend color for bar objects.

Syntax

```c
BOOL SetTrendColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lTrendColor);
```

Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

- `lTrendColor`
  Numeric value defining the trend color
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetUnselBGColor

Function
Sets the background color of non-selected entries for the "text list" object.

Syntax

BOOL SetUnselBGColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lUnselBGColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lUnselBGColor
Numeric value defining the background color for non-selected entries

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetUnselTextColor

Function

Sets the font color of non-selected entries for the "text list" object.

Syntax

BOOL SetUnselTextColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lUnselTextColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lUnselTextColor
Numeric value defining the font color for non-selected entries

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
fill

Fill - short description
The functions in the Fill group control the dynamic filling of objects.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetFilling

Function
Activates or deactivates dynamic filling with background color.

Syntax
BOOL SetFilling(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFilling);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bFilling
Dynamic filling with background color on/off

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetFilling example (Page 1561)
SetFillingIndex

Function

Sets the fill level.

Syntax

BOOL SetFillingIndex(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillingIndex);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lFillingIndex
Fill level as a numeric value (0 - 100)

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetFillingIndex example (Page 1561)
flash

Flash - short description
The various flashing properties can be modified or called in using the functions in the Flash group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetBackFlashColorOff

Function
Sets the background flash color for the deactivated status.

Syntax
BOOL SetBackFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackFlashColorOff);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lBackFlashColorOff
Background flash color for the deactivated status as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

Color chart (Page 1592)
SetFlashBackColorOn example (Page 1562)

SetBackFlashColorOn

Function

Sets the background flash color for the activated status.

Syntax

BOOL SetBackFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackFlashColorOn);

Parameters

*lpszPictureName*
Picture name

*lpszObjectName*
Object name

*lBackFlashColorOn*
Background flash color for the activated status as a numeric value

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

Color chart (Page 1592)
SetFlashBackColorOn example (Page 1562)
SetBorderFlashColorOff

Function
Sets the border or line flashing color for the deactivated status.

Syntax
BOOL SetBorderFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderFlashColorOff);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lBorderFlashColorOff
Border or line flashing color for the deactivated status as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetFlashBackColorOn example (Page 1562)
Color chart (Page 1592)

SetBorderFlashColorOn

Function
Sets the border or line flashing color for the activated status.

Syntax
BOOL SetBorderFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderFlashColorOn);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lBorderFlashColorOn**
  Border or line flashing color for the activated status as a numeric value

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Color chart (Page 1592)
- SetFlashBackColorOn example (Page 1562)

SetFlashBackColor

Function

Activates or deactivates background flashing.

Syntax

```c
BOOL SetFlashBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashBackColor);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **bFlashBackColor**
  Flashing background on/off
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetFlashBackColor example (Page 1562)

SetFlashBorderColor

Function
Activates or deactivates flashing of the border or line.

Syntax
BOOL SetFlashBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashBorderColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bFlashBorderColor
Flashing of the border or line on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetFlashBackColor example (Page 1562)
SetFlashForeColor

Function
Activates or deactivates font flashing.

Syntax
BOOL SetFlashForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashForeColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bFlashForeColor
Flashing of the font on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetFlashBackColor example (Page 1562)

SetFlashRateBackColor

Function
Sets the flash frequency of the background.

Syntax
BOOL SetFlashRateBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateBackColor);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lFlashRateBackColor**
  Flash frequency of the background

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Flash frequencies (Page 1590)
- SetFlashBackColor example (Page 1562)

**SetFlashRateBorderColor**

Function

Sets the flash frequency of the line or border.

Syntax

```c
BOOL SetFlashRateBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateBorderColor);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lFlashRateBorderColor**
  Flash frequency of the line or border
SetFlashRateForeColor

Function

Sets the flash frequency of the font.

Syntax

BOOL SetFlashRateForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateForeColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lFlashRateForeColor
Flash frequency of the font

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Flash frequencies (Page 1590)
SetFlashBackColor example (Page 1562)
See also

Flash frequencies (Page 1590)
SetFlashBackColor example (Page 1562)

SetForeFlashColorOff

Function

Sets the font flash color for the deactivated status.

Syntax

BOOL SetForeFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeFlashColorOff);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lForeFlashColorOff

Font flash color for the deactivated status as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetFlashBackColorOn example (Page 1562)
SetForeFlashColorOn

Function

Sets the font flash color for the activated status.

Syntax

BOOL SetForeFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeFlashColorOn);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lForeFlashColorOn
Font flash color for the activated status as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetFlashBackColorOn example (Page 1562)
Focus - short description

Using the functions in the Focus group, it is possible to set the focus or poll which object has the focus.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

Set_Focus

Function

Sets the focus on the specified object.

Syntax

BOOL Set_Focus(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- SetFocus example (Page 1563)
font

Font - short description
The various properties affecting text can be modified or called in using the functions in the Font group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetAlignmentLeft

Function
Sets the horizontal text alignment (left, centered, right).

Syntax
BOOL SetAlignmentLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lAlignmentLeft);

Parameters

  lpszPictureName
  Picture name

  lpszObjectName
  Object name

  lAlignmentLeft
  Horizontal text alignment as a numeric value

Return value

  TRUE
  The function has been completed without any errors.

  FALSE
  An error has occurred.
See also

Text alignment (Page 1597)
SetFontSize example (Page 1564)

SetAlignmentTop

Function

Sets the vertical text alignment (top, centered, bottom).

Syntax

BOOL SetAlignmentTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lAlignmentTop);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lAlignmentTop
Vertical text alignment as a numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Text alignment (Page 1597)
SetFontSize example (Page 1564)
SetFontBold

Function

Switches the bold font on or off.

Syntax

BOOL SetFontBold(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontBold);

Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

bFontBold
Bold font on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetFontBold example (Page 1563)

SetFontItalic

Function

Switches the italic font on or off.

Syntax

BOOL SetFontItalic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontItalic);
Parameters

\texttt{lpszPictureName}
Picture name

\texttt{lpszObjectName}
Object name

\texttt{bFontItalic}
Italic font on/off

Return value

\texttt{TRUE}
The function has been completed without any errors.

\texttt{FALSE}
An error has occurred.

See also

SetFontBold example (Page 1563)

SetFontName

Function
Sets a font.

Syntax

\texttt{BOOL SetFontName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szFontName);}
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetText example (Page 1578)

SetFontSize

Function
Sets the font size.

Syntax

BOOL SetFontSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFontSize);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lFontSize
Font Size

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetFontSize example (Page 1564)
SetFontUnderline

Function

Switches the underlined font on or off.

Syntax

BOOL SetFontUnderline(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontUnderline);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bFontUnderline
Underlined font on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetFontBold example (Page 1563)

SetOrientation

Function

Defines the text orientation (vertical/horizontal).

Syntax

BOOL SetOrientation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOrientation);
Parameters

**lpzPictureName**
Picture name

**lpzObjectName**
Object name

**bOrientation**
Text orientation

| TRUE  | vertical |
| FALSE | Horizontal |

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

SetFontBold example (Page 1563)

SetText

Function

Sets the value of the “text” property for objects like static text, check box or radio box.

Syntax

```c
BOOL SetText(LPCTSTR lpzPictureName, LPCTSTR lpzObjectName, char* szText);
```

Parameters

**lpzPictureName**
Picture name

**lpzObjectName**
Object name
szText
Pointer to a text

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
In case of check and radio boxes the element to be changed must be defined with the SetIndex function before actually activating this function.

See also
SetText example (Page 1578)

gometry

Geometry - short description
The size, position and other geometrical properties of objects can be modified or called in using the functions in the Geometry group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetActualPointLeft

Function
Sets the X value for the current point of a polygon or polygon line.

Syntax
BOOL SetActualPointLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lActualPointLeft);
Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
- **lActualPointLeft**: X value for the current point of a polygon or polygon line

Return value

- **TRUE**: The function has been completed without any errors.
- **FALSE**: An error has occurred.

Note

The current point of the polygon can be set using the SetIndex function.

See also

SetLeft example (Page 1565)

**SetActualPointTop**

Function

Sets the Y value for the current point of a polygon or polygon line.

Syntax

```c
BOOL SetActualPointTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lActualPointTop);
```

Parameters

- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
IActualPointTop
Y value for the current point of a polygon or polygon line

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
The current point of the polygon can be set using the SetIndex function.

See also
SetTop example (Page 1578)

SetBoxCount
Function
Sets the number of fields in a check box or radio box.

Syntax
BOOL SetBoxCount(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxCount);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lBoxCount
Number of fields in a check box or radio box.
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

SetDirection

Function

Sets the bar direction for bar objects.

Syntax

```c
BOOL SetDirection(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lDirection);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lDirection**
  Numeric value defining the bar direction

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

- Bar direction (Page 1590)
- SetTop example (Page 1578)
SetEndAngle

Function
Sets the end angle of circle and ellipse segments and circle and elliptical arcs.

Syntax
BOOL SetEndAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lEndAngle);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lEndAngle**
  End angle of circle and ellipse segments as well as circle and ellipse arcs

Return value
- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

See also
SetTop example (Page 1578)

SetHeight

Function
Sets the height of the rectangle framing an object.

Syntax
BOOL SetHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int IHeight);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lHeight**
  Height of the framing rectangle

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

Note

If the function is called in relation to the entire picture, set the parameter `lpszObjectName = NULL`.

See also

- SetHeight example (Page 1564)

SetLeft

Function

Sets the X value of the upper left corner of the rectangle framing an object

Syntax

```c
BOOL SetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLeft);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name
**ILeft**

X value of the upper left corner of the framing rectangle

**Return value**

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

**See also**

SetLeft example (Page 1565)

**SetPointCount**

**Function**

Sets the number of corners of a polygon or in a polygon line.

**Syntax**

```c
BOOL SetPointCount(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPointCount);
```

**Parameters**

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lPointCount**
  Number of corner points

**Return value**

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.
See also

SetLeft example (Page 1565)

SetRadius

Function

Sets the radius of a circle, circle segment or arc.

Syntax

BOOL SetRadius(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadius);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lRadius
Radius

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHeight example (Page 1564)

SetRadiusHeight

Function

Sets the radius of an ellipse, ellipse segment or elliptical arc in vertical direction.
Syntax

BOOL SetRadiusHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadiusHeight);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lRadiusHeight
Radius in vertical direction

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHeight example (Page 1564)

SetRadiusWidth

Function

Sets the radius of an ellipse, ellipse segment or elliptical arc in horizontal direction.

Syntax

BOOL SetRadiusWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadiusWidth);

Parameters

lpszPictureName
Picture name
SetReferenceRotationLeft

Function

Sets the X value of the rotation reference (central axis about which the object can be rotated) for lines, polygons and polylines.

Syntax

BOOL SetReferenceRotationLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int IReferenceRotationLeft);

Parameters

IpszPictureName
Picture name

IpszObjectName
Object name

IReferenceRotationLeft
X value of the rotation reference
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetLeft example (Page 1565)

SetReferenceRotationTop

Function
Sets the Y value of the rotation reference (central axis about which the object can be rotated) for lines, polygons and polylines.

Syntax
BOOL SetReferenceRotationTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lReferenceRotationTop);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lReferenceRotationTop
Y value of the rotation reference

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

SetTop example (Page 1578)

SetRotationAngle

Function

Sets the angle of rotation about the central axis for lines, polygons and polylines.

Syntax

BOOL SetRotationAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRotationAngle);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lRotationAngle
Angle of rotation

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetLeft example (Page 1565)

SetRoundCornerHeight

Function

Specifies the radius of the rounded corner of a rectangle vertically.
Syntax

BOOL SetRoundCornerHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRoundCornerHeight);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lRoundCornerHeight
Vertical radius

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHeight example (Page 1564)

SetRoundCornerWidth

Function
Specifies the radius of the rounded corner of a rectangle horizontally.

Syntax

BOOL SetRoundCornerWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRoundCornerWidth);

Parameters

lpszPictureName
Picture name
IpszObjectName
Object name

IRoundCornerWidth
Horizontal radius

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetWidth example (Page 1579)

SetStartAngle

Function
Sets the start angle of circle and ellipse segments and circle and elliptical arcs.

Syntax

BOOL SetStartAngle(LPCTSTR IpszPictureName, LPCTSTR IpszObjectName, long int IStartAngle);

Parameters

IpszPictureName
Picture name

IpszObjectName
Object name

IStartAngle
Starting angle

Return value

TRUE
The function has been completed without any errors.
SetTop

Function
Sets the Y value of the upper left corner of the rectangle framing an object.

Syntax
BOOL SetTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int ITop);

Parameters
- lpszPictureName
  Picture name
- lpszObjectName
  Object name
- ITop
  Y value of the upper left corner of the framing rectangle

Return value
- TRUE
  The function has been completed without any errors.
- FALSE
  An error has occurred.

See also
- SetHeight example (Page 1564)
- SetTop example (Page 1578)
SetWidth

Function
Sets the width of the rectangle framing an object.

Syntax
BOOL SetWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lWidth);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lWidth**
  Width of the framing rectangle

Return value
- **TRUE**
The function has been completed without any errors.
- **FALSE**
  An error has occurred.

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also
SetWidth example (Page 1579)

SetZeroPoint

Function
Sets the zero point for bar objects.
Syntax

BOOL SetZeroPoint(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lZeroPoint);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lZeroPoint
Zero point

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetTop example (Page 1578)

i_o

i_o - short description

The various properties affecting input and output values can be modified or called in using the functions in the i_o group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
SetAssumeOnExit

Function

Specifies for I/O fields whether the entered value is assumed upon exiting the field.

Syntax

BOOL SetAssumeOnExit(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAssumeOnExit);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bAssumeOnExit
Value application upon exiting the field yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHiddenInput example (Page 1564)

SetAssumeOnFull

Function

Specifies for I/O fields whether the entered value is assumed on completion of input.

Syntax

BOOL SetAssumeOnFull(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAssumeOnFull);
Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

_bAssumeOnFull
Value application on completion of input yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetHiddenInput example (Page 1564)

SetBitNumber

Function
Sets the relevant bit in the output value for the "bit" list type.

Syntax

BOOL SetBitNumber(LPCTSTR _lpszPictureName, LPCTSTR _lpszObjectName, long _lBitNumber);

Parameters

_lpszPictureName
Picture name

_lpszObjectName
Object name

_lBitNumber
Relevant bit in the output value for the "bit" list type
SetClearOnError

Function

Specifies for I/O fields whether deletion of the content in case of input errors is activated.

Syntax

BOOL SetClearOnError(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bClearOnError);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bClearOnError
Deletion of the entry in case of input errors yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHiddenInput example (Page 1564)
SetClearOnNew

Function

Specifies the deletion of the content in case of new inputs for I/O fields.

Syntax

BOOL SetClearOnNew(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bClearOnNew);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bClearOnNew
Deletion of content in case of new input yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHiddenInput example (Page 1564)

SetHiddenInput

Function

Controls the hidden input for I/O fields.

Syntax

BOOL SetHiddenInput(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bHiddenInput);
Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bHiddenInput
Hidden input yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetHiddenInput example (Page 1564)

SetNumberLines

Function

Sets the number of visible lines lines for the "text list" object.

Syntax

BOOL SetNumberLines(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lNumberLines);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lNumberLines
Number of visible lines
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the amount of configured text is larger than the number of visible lines, the "text list" object receives a vertical scroll bar.

SetOutputValueChar

Function
Sets a pointer to the output value for I/O fields

Syntax
BOOL SetOutputValueChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szOutputValueChar);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

szOutputValueChar
Pointer to the output value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetOutputValueDouble

Function

Sets the output value for I/O fields.

Syntax

BOOL SetOutputValueDouble(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dOutputValueDouble);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dOutputValueDouble
Output value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetOutputValueDouble example (Page 1566)
Limits

Limits - short description

The various properties affecting limit values can be modified or called in using the functions in the Limits group.

---

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

---

SetAlarmHigh

Function

Sets the upper alarm limit for bar objects.

Syntax

BOOL SetAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAlarmHigh);

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **dAlarmHigh**
  Upper alarm limit

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.
See also

SetAlarmHigh example (Page 1559)

SetAlarmLow

Function

Sets the lower alarm limit for bar objects.

Syntax

BOOL SetAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAlarmLow);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dAlarmLow
Lower alarm limit

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetAlarmHigh example (Page 1559)

SetCheckAlarmHigh

Function

Controls the monitoring of the upper alarm limit for bar objects.
Syntax

BOOL SetCheckAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckAlarmHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bCheckAlarmHigh
Monitoring yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetCheckAlarmLow

Function

Controls the monitoring of the lower alarm limit for bar objects.

Syntax

BOOL SetCheckAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckAlarmLow);

Parameters

lpszPictureName
Picture name
SetCheckLimitHigh4

Function
Controls the monitoring of the upper limit value reserve 4 for bar objects.

Syntax
BOOL SetCheckLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitHigh4);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **bCheckLimitHigh4**
  Monitoring yes/no

Return value
- **TRUE**
The function has been completed without any errors.
- **FALSE**
  An error has occurred.

See also
SetMarker example (Page 1566)
FALSE
An error has occurred.

See also
SetMarker example (Page 1566)

SetCheckLimitHigh5

Function
Controls the monitoring of the upper limit value reserve 5 for bar objects.

Syntax
BOOL SetCheckLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitHigh5);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

bCheckLimitHigh5
Monitoring yes/no

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetMarker example (Page 1566)
SetCheckLimitLow4

Function
Controls the monitoring of the lower limit value reserve 4 for bar objects.

Syntax
BOOL SetCheckLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitLow4);

Parameters
- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **bCheckLimitLow4**
  Monitoring yes/no.

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also
SetMarker example (Page 1566)

SetCheckLimitLow5

Function
Controls the monitoring of the lower limit value reserve 5 for bar objects.

Syntax
BOOL SetCheckLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitLow5);
Parameters

$lpszPictureName$
Picture name

$lpszObjectName$
Object name

$bCheckLimitLow$
Monitoring yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetCheckToleranceHigh

Function

Controls the monitoring of the upper tolerance limit for bar objects.

Syntax

BOOL SetCheckToleranceHigh(LPCTSTR $lpszPictureName$, LPCTSTR $lpszObjectName$, BOOL $bCheckToleranceHigh$);

Parameters

$lpszPictureName$
Picture name

$lpszObjectName$
Object name

$bCheckToleranceHigh$
Monitoring yes/no
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetMarker example (Page 1566)

SetCheckToleranceLow

Function
Controls the monitoring of the lower tolerance limit for bar objects.

Syntax
BOOL SetCheckToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckToleranceLow);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bCheckToleranceLow
Monitoring yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetMarker example (Page 1566)
SetCheckWarningHigh

Function

Controls the monitoring of the upper warning limit for bar objects.

Syntax

BOOL SetCheckWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckWarningHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bCheckWarningHigh
Monitoring yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetCheckWarningLow

Function

Controls the monitoring of the lower warning limit for bar objects.

Syntax

BOOL SetCheckWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckWarningLow);
Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bCheckWarningLow
Monitoring yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetMarker example (Page 1566)

SetColorAlarmHigh

Function
Sets the bar color for bar objects upon reaching the upper alarm limit.

Syntax

BOOL SetColorAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorAlarmHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lColorAlarmHigh
Numeric value defining the bar color
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetColorAlarmLow example (Page 1559)

SetColorAlarmLow

Function

Sets the bar color for bar objects upon reaching the lower alarm limit.

Syntax

BOOL SetColorAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int IColorAlarmLow);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

IColorAlarmLow
Numeric value defining the bar color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetColorLimitHigh4

Function

Sets the bar color for bar objects upon reaching the upper limit reserve 4.

Syntax

BOOL SetColorLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitHigh4);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lColorLimitHigh4
Numeric value defining the bar color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
SetColorLimitHigh5

Function

Sets the bar color for bar objects upon reaching the upper limit reserve 5.

Syntax

BOOL SetColorLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitHigh5);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

IColorLimitHigh5
Numeric value defining the bar color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetColorBack example (Page 1559)

SetColorLimitLow4

Function

Sets the bar color for bar objects upon reaching the lower limit reserve 4.

Syntax

BOOL SetColorLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitLow4);
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lColorLimitLow4**
  Numeric value defining the bar color

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetColorLimitLow5

Function

Sets the bar color for bar objects upon reaching the lower limit reserve 5.

Syntax

```c
BOOL SetColorLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitLow5);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lColorLimitLow5**
  Numeric value defining the bar color
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

Color chart (Page 1592)
SetColorToleranceHigh example (Page 1559)

SetColorToleranceHigh

Function

Sets the bar color for bar objects upon reaching the upper tolerance limit.

Syntax

```c
BOOL SetColorToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorToleranceHigh);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lColorToleranceHigh**
  Numeric value defining the bar color

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.
See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetColorToleranceLow

Function

Sets the bar color for bar objects upon reaching the lower tolerance limit.

Syntax

BOOL SetColorToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int iColorToleranceLow);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

IColorToleranceLow
Numeric value defining the bar color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)
SetColorWarningHigh

Function

Sets the bar color for bar objects upon reaching the upper warning limit.

Syntax

BOOL SetColorWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorWarningHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lColorWarningHigh
Numeric value defining the bar color

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetColorWarningLow

Function

Sets the bar color for bar objects upon reaching the lower warning limit.

Syntax

BOOL SetColorWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorWarningLow);
Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

- `IColorWarningLow`
  Numeric value defining the bar color

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetLimitHigh4

Function

Sets the high limit value for reserve 4 for bar objects.

Syntax

```c
BOOL SetLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitHigh4);
```

Parameters

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

- `dLimitHigh4`
  High limit value for reserve 4
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetAlarmHigh example (Page 1559)

SetLimitHigh5

Function
Sets the high limit value for reserve 5 for bar objects.

Syntax
BOOL SetLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitHigh5);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dLimitHigh5
High limit value for reserve 5

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetAlarmHigh example (Page 1559)
SetLimitLow4

Function
Sets the low limit value for reserve 4 for bar objects.

Syntax
BOOL SetLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitLow4);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

dLimitLow4
Low limit value for reserve 4

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetAlarmHigh example (Page 1559)

SetLimitLow5

Function
Sets the low limit value for reserve 5 for bar objects.

Syntax
BOOL SetLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitLow5);
SetLimitMax

Function

Sets the high limit value for I/O fields.

Syntax

```c
BOOL SetLimitMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitMax);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **dLimitMax**
  High limit value

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- SetAlarmHigh example (Page 1559)
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetAlarmHigh example (Page 1559)

SetLimitMin

Function
Sets the low limit value for I/O fields.

Syntax
BOOL SetLimitMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitMin);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

dLimitMin
Lower limit

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetAlarmHigh example (Page 1559)
SetMarker

Function

Controls the limit marker display for bar objects.

Syntax

BOOL SetMarker(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bMarker);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bMarker
Limit marker on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetToleranceHigh

Function

Sets the upper tolerance limit for bar objects.

Syntax

BOOL SetToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dToleranceHigh);
Parameters

*lpszPictureName*
Picture name

*lpszObjectName*
Object name

*dToleranceHigh*
Upper tolerance limit

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

SetAlarmHigh example (Page 1559)

SetToleranceLow

Function

Sets the lower tolerance limit for bar objects.

Syntax

```c
BOOL SetToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dToleranceLow);
```

Parameters

*lpszPictureName*
Picture name

*lpszObjectName*
Object name

*dToleranceLow*
Lower tolerance limit
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetAlarmHigh example (Page 1559)

SetTypeAlarmHigh

Function

Specifies for bar objects whether the upper alarm limit is given in percentages or absolute terms.

Syntax

BOOL SetTypeAlarmHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeAlarmHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bTypeAlarmHigh
Upper alarm limit

TRUE Specification in percent
FALSE Absolute specification

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

SetMarker example (Page 1566)

**SetTypeAlarmLow**

**Function**

Specifies for bar objects whether the lower alarm limit is given in percentages or absolute terms.

**Syntax**

```c
BOOL SetTypeAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeAlarmLow);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **bTypeAlarmLow**
  Lower alarm limit
  
<table>
<thead>
<tr>
<th>TRUE</th>
<th>Specification in percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>Absolute specification</td>
</tr>
</tbody>
</table>

**Return value**

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

SetMarker example (Page 1566)
SetTypeLimitHigh4

Function

Specifies for bar objects whether the upper limit for reserve 4 is given in percentages or absolute terms.

Syntax

BOOL SetTypeLimitHigh4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitHigh4);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bTypeLimitHigh4
High limit

TRUE Specification in percent
FALSE Absolute specification

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetTypeLimitHigh5

Function

Specifies for bar objects whether the upper limit for reserve 5 is given in percentages or absolute terms.
Syntax

```c
BOOL SetTypeLimitHigh5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitHigh5);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **bTypeLimitHigh5**
  High limit
  - TRUE Specification in percent
  - FALSE Absolute specification

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- SetMarker example (Page 1566)

SetTypeLimitLow4

Function

Specifies for bar objects whether the lower limit for reserve 4 is given in percentages or absolute terms.

Syntax

```c
BOOL SetTypeLimitLow4(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitLow4);
```
Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**bTypeLimitLow4**
Low limit

- TRUE  Specification in percent
- FALSE Absolute specification

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

SetMarker example (Page 1566)

**SetTypeLimitLow5**

Function

Specifies for bar objects whether the lower limit for reserve 5 is given in percentages or absolute terms.

Syntax

```c
BOOL SetTypeLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitLow5);
```

Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name
bTypeLimitLow
Low limit

TRUE Specification in percent
FALSE Absolute specification

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetMarker example (Page 1566)

SetTypeToleranceHigh

Function
Specifies for bar objects whether the high tolerance limit is given in percentages or absolute terms.

Syntax
BOOL SetTypeToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeToleranceHigh);

Parameter

lpszPictureName
Picture name

lpszObjectName
Object name

bTypeToleranceHigh
High tolerance limit

TRUE Specification in percent
FALSE Absolute specification
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)

SetTypeToleranceLow

Function

Specifies for bar objects whether the lower tolerance limit is given in percentages or absolute terms.

Syntax

BOOL SetTypeToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeToleranceLow);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bTypeToleranceLow
Lower tolerance limit

| TRUE   | Specification in percent |
| FALSE  | Absolute specification  |

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

SetMarker example (Page 1566)

SetTypeWarningHigh

Function

Specifies for bar objects whether the upper warning limit is given in percentages or absolute terms.

Syntax

BOOL SetTypeWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeWarningHigh);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bTypeWarningHigh
Upper warning limit

    TRUE  Specification in percent
    FALSE  Absolute specification

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetMarker example (Page 1566)
**SetTypeWarningLow**

**Function**
Specifies for bar objects whether the lower warning limit is given in percentages or absolute terms.

**Syntax**

```c
BOOL SetTypeWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeWarningLow);
```

**Parameters**

- `lpszPictureName`
  Picture name

- `lpszObjectName`
  Object name

- `bTypeWarningLow`
  Lower warning limit

  - **TRUE** Specification in percent
  - **FALSE** Absolute specification

**Return value**

- **TRUE** The function has been completed without any errors.
- **FALSE** An error has occurred.

**See also**
SetMarker example (Page 1566)

**SetWarningHigh**

**Function**
Sets the upper warning limit for bar objects.
**BOOL** SetWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dWarningHigh);

### Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **dWarningHigh**
  Upper warning limit

### Return value

- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

### See also

SetMarker example (Page 1566)

---

**BOOL** SetWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dWarningLow);

### Function

Sets the lower warning limit for bar objects.

### Syntax

- **lpszPictureName**
  Picture name
The function has been completed without any errors.

An error has occurred.

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lpszPropertyName**
  Name of the object property

- **pLink**
  Pointer to a structure of the type: LINKINFO

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

See also

- Structure definition LINKINFO (Page 1605)
- SetLink example (Page 1565)

miscs

**Miscs - short description**

The properties of objects can be modified or called in using the functions in the Miscs group.

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

**SetAverage**

**Function**

Controls the averaging of bar objects.
Syntax

BOOL SetAverage(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAverage);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bAverage
Averaging yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetVisible example (Page 1578)

SetBoxType

Function

Specifies the field type (input field, output field, input/output field) for an I/O object.

Syntax

BOOL SetBoxType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxType);

Parameters

lpszPictureName
Picture name
SetColorChangeType

Function

When using bar objects, it defines whether the color change upon reaching a limit value only affects a bar segment or the entire bar.

Syntax

BOOL SetColorChangeType(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bColorChangeType);

Parameter

lpszPictureName
Picture name

lpszObjectName
Object name

bColorChangeType
Type of color change

TRUE Color change applies to a segment
FALSE Color change applies to the entire bar

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

I/O field, field type (Page 1592)
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetVisible example (Page 1578)

SetCursorControl

Function
Sets the cursor control for I/O fields.

Syntax
BOOL SetCursorControl(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCursorControl);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bCursorControl
Cursor control on/off

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetCursorMode

Function
Sets the cursor control for pictures.

Syntax
BOOL SetCursorMode(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCursorMode);

Parameters
lpszPictureName
Picture name

lpszObjectName
Object name

bCursorMode
Cursor Mode
   TRUE   Tab order cursor
   FALSE  Alpha-Cursor

Return value
TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
Example - SetCursorMode (Page 1561)
SetEditAtOnce

Function

Specifies whether the "Immediate input" property is activated for I/O fields.

Syntax

BOOL SetEditAtOnce(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bEditAtOnce);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bEditAtOnce
Immediate input yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetVisible example (Page 1578)

SetExtendedOperation

Function

Controls the "Extended operation" property of slider objects.

Syntax

BOOL SetExtendedOperation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bExtendedOperation);
Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**bExtendedOperation**
Extended operation yes/no

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

SetVisible example (Page 1578)

SetHysteresis

Function

When using bar objects, it specifies whether the display appears with or without hysteresis.

Syntax

```c
BOOL SetHysteresis(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bHysteresis);
```

Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**bHysteresis**
Display with/without hysteresis
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetVisible example (Page 1578)

SetHysteresisRange

Function

Sets the hysteresis value in the display for bar objects.

Syntax

BOOL SetHysteresisRange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dHysteresisRange);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dHysteresisRange
Hysteresis value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetMax

Function
Sets the maximum value for bar and slider objects.

Syntax
BOOL SetMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dMax);

Parameters
- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
- **dMax**: Maximum value

Return value
- **TRUE**: The function has been completed without any errors.
- **FALSE**: An error has occurred.

SetMin

Function
Sets the minimum value for bar and slider objects.

Syntax
BOOL SetMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dMin);

Parameters
- **lpszPictureName**: Picture name
SetOffsetLeft

Function
Sets the horizontal picture distance from the left window border for picture windows.

Syntax

```c
BOOL SetOffsetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lOffsetLeft);
```

Parameters

- `lpszPictureName`: Picture name
- `lpszObjectName`: Object name
- `lOffsetLeft`: Picture distance

Return value

- **TRUE**: The function has been completed without any errors.
- **FALSE**: An error has occurred.
SetOffsetTop

Function
Sets the vertical picture distance from the upper window border for picture windows.

Syntax
BOOL SetOffsetTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lOffsetTop);

Parameters
- **lpszPictureName**: Picture name
- **lpszObjectName**: Object name
- **lOffsetTop**: Picture distance

Return value
- **TRUE**: The function has been completed without any errors.
- **FALSE**: An error has occurred.

SetOperation

Function
Controls the operability of the objects.

Syntax
BOOL SetOperation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperation);
Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**bOperation**
Object operable, yes/no

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

---

**Note**
If the function is called for the picture object, set the parameter lpszObjectName = NULL.

---

**See also**
SetVisible example (Page 1578)

---

**SetOperationMessage**

**Function**
Controls the output of a message when operating the objects "I/O field", "Check box", "Radio box" and "Slider".

**Syntax**

```c
BOOL SetOperationMessage(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperationMessage);
```

**Parameters**

**lpszPictureName**
Picture name
SetOperationReport

Function

Controls the logging of the operating reason for all objects except application and picture windows and OLE control.

Syntax

BOOL SetOperationReport(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperationReport);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bOperationReport
Logging operating reason yes/no
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

---

**Note**
If the function is called for the picture object, set the parameter lpszObjectName = NULL.

See also

SetVisible example (Page 1578)

---

**SetPasswordLevel**

**Function**
Defines the authorization level for operating objects for all objects except application and picture windows and OLE control.

**Syntax**

```c
BOOL SetPasswordLevel(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPasswordLevel);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lPasswordLevel**
  Authorization level

**Return value**

**TRUE**
The function has been completed without any errors.
**FALSE**
An error has occurred.

---

**Note**
If the function is called in relation to the entire picture, set the parameter `lpszObjectName = NULL`.

---

**SetPictureName**

**Function**
Sets the name of the picture, which should be shown in a picture window or in a graphic object.

**Syntax**

```c
BOOL SetPictureName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureName);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Name of the picture window or graphic object

- **szPictureName**
  Pointer to the picture name

**Return value**

- **TRUE**
The function has been completed without any errors.

- **FALSE**
An error has occurred.

**See also**

SetPictureName example (Page 1567)
SetProcess

Function

Specifies the default setting of the value to be displayed for bar and slider objects.
Sets the selected fields for check boxes and radio boxes.

Syntax

BOOL SetProcess(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dProcess);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dProcess

- In case of bar and slider objects, this value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.
- For check boxes and radio boxes the selected fields are specified. In the 32-bit word each field is represented by a bit (field 1 corresponds to the bit value 0). Selected fields are marked by a set bit. Non-existing are assigned 0.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

SetSmallChange

Function

Sets the number of steps for slider objects by which the slider is shifted by a mouse click.

Syntax

BOOL SetSmallChange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSmallChange);
Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name

**lSmallChange**
Number of setting steps

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

SetTagPrefix

Function

This function sets the tag prefix of a picture window:

In a picture window the "temperature" tag is requested on an object. If a "Motor1." tag prefix is assigned to the picture window, the tag "Motor1.Temperature" is requested.

The setting of the tag prefix only becomes effective when newly supplying the picture name.

This means you must either set the prefix before picture selection or newly supply the picture name if the picture is not changed.

Syntax

```c
BOOL SetTagPrefix(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szTagPrefix);
```

Parameters

**lpszPictureName**
Picture name

**lpszObjectName**
Object name
szTagPrefix
Tag prefix to be set

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the tag prefix is set for a picture window, the tag prefix is added to all tags contained in the picture to be displayed. This also applies if the request takes place in a function. If a tag needs to be read without the tag prefix, you must add "@NOTP::" to the tag name.

Using a tag prefix greatly simplifies the picture module technology.

See also
SetTagPrefix example (Page 1574)

SetTrend

Function
Controls the trend display for bar objects.

Syntax
BOOL SetTrend(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTrend);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bTrend
Trend display yes/no
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

SetVisible example (Page 1578)

SetVisible

Function

Controls the display of an object.

Syntax

```c
BOOL SetVisible(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bVisible);
```

Parameters

- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **bVisible**
  Object display yes/no

Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.
See also

SetVisible example (Page 1578)

SetZeroPointValue

Function

Sets the absolute value of the zero point for bar objects.

Syntax

BOOL SetZeroPointValue(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dZeroPointValue);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

dZeroPointValue
Absolute value of the zero point

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

SetZoom

Function

Sets the scaling factor for a picture window.

Syntax

BOOL SetZoom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lZoom);
Parameters

`lpszPictureName`
Picture name

`lpszObjectName`
Object name

`lZoom`
Scaling factor

Return value

`TRUE`
The function has been completed without any errors.

`FALSE`
An error has occurred.

ole_control

**OLE_control - short description**

The functions in the ole_Control group can only be used with OCX slider objects.

Various OCX slider object properties and settings can be modified or called in using these functions.

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetPosition

**Function**

Sets the slider position of the OCX slider object.

**Syntax**

```c
BOOL SetPosition(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPosition);
```
Parameters

\textbf{lpszPictureName}  
Picture name

\textbf{lpszObjectName}  
Object name

\textbf{lPosition}  
Slider position of the OCX slider object

Return value

\textbf{TRUE}  
The function has been completed without any errors.

\textbf{FALSE}  
An error has occurred.

See also

SetPosition example (Page 1568)

\textbf{SetRangeMax}

Function

Defines the adjustment range "Max" of the OCX slider object.

Syntax

\texttt{BOOL SetRangeMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRangeMax);}  

Parameters

\textbf{lpszPictureName}  
Picture name

\textbf{lpszObjectName}  
Object name

\textbf{lRangeMax}  
Adjustment range "Max" of the OCX slider object
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetRangeMax example (Page 1569)

SetRangeMin

Function
Defines the adjustment range "Min" of the OCX slider object.

Syntax

BOOL SetRangeMin(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRangeMin);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lRangeMin
Adjustment range "Min" of the OCX slider object

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

SetRangeMin example (Page 1569)
pictures

Pictures - short description

Various properties of pictures of graphic objects and round buttons can be modified or called in using the functions in the Pictures group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetPicDeactTransparent

Function

Sets the transparent color for the "deactivated" status of a round button.

Syntax

BOOL SetPicDeactTransparent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicDeactTransparent);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lPicDeactTransparent
Transparent color for "deactivated" status

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
This function only applies to Bitmap graphics (*.bmp).

See also
Color chart (Page 1592)
SetBackColor example (Page 1559)

SetPicDeactUseTransColor

Function
Controls the transparent color for the "deactivated" status of a round button.

Syntax
BOOL SetPicDeactUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicDeactUseTransColor);

Parameter

lpszPictureName  Picture name

lpszObjectName  Object name

bPicDeactUseTransColor  Transparent color yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetPicDownTransparent

Function
Sets the transparent color for the "On/pressed" status of a round button.

Syntax
BOOL SetPicDownTransparent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicDownTransparent);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **lPicDownTransparent**
  Transparent color for "On/pressed" status

Return value
- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

Note
This function only applies to Bitmap graphics (*.bmp).

See also
- Color chart (Page 1592)
- SetBackColor example (Page 1559)

SetPicDownUseTransColor

Function
Controls the transparent color for the "On/pressed" status of a round button.
Syntax

```c
BOOL SetPicDownUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicDownUseTransColor);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **bPicDownUseTransColor**
  Transparent color yes/no

Return value

- **TRUE**
  The function has been completed without any errors.

- **FALSE**
  An error has occurred.

SetPicTransColor

Function

Sets the transparent color of the background picture of a graphic object.

Syntax

```c
BOOL SetPicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicTransColor);
```

Parameters

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lPicTransColor**
  Transparent color of the background picture
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

---

**Note**
This function only applies to Bitmap graphics (*.bmp).

---

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

---

SetPictureDeactivated

**Function**
Specifies the picture name for the "deactivated" status of a round button.

**Syntax**

```c
BOOL SetPictureDeactivated(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureDeactivated);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszPictureName</td>
<td>Picture name</td>
</tr>
<tr>
<td>lpszObjectName</td>
<td>Object name</td>
</tr>
<tr>
<td>szPictureDeactivated</td>
<td>Picture name for &quot;deactivated&quot; status</td>
</tr>
</tbody>
</table>

**Return value**

**TRUE**
The function has been completed without any errors.
FALSE
An error has occurred.

Note
Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also
SetPictureDown example (Page 1567)

SetPictureDown

Function
Specifies the picture name for the "On/pressed" status of a round button.

Syntax
BOOL SetPictureDown(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureDown);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

szPictureDown
Picture name for "On/pressed" status

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.
See also
SetPictureDown example (Page 1567)

SetPictureUp

Function
Specifies the picture name for the "Off/not pressed" status of a round button.

Syntax
BOOL SetPictureUp(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureUp);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

szPictureUp
Picture name for "Off/not pressed" status

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also
SetPictureUp example (Page 1567)
SetPicUpTransparent

Function

Sets the transparent color for the "Off/not pressed" status of a round button.

Syntax

BOOL SetPicUpTransparent(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int IPicUpTransparent);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

IPicUpTransparent
Transparent color for "Off/not pressed" status

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
This function only applies to Bitmap graphics (*.bmp).

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

SetPicUpUseTransColor

Function

Controls the transparent color for the "Off/not pressed" status of a round button.
Syntax

```c
BOOL SetPicUpUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName,
BOOL bPicUpUseTransColor);
```

Parameters

- **lpszPictureName**
  - Picture name

- **lpszObjectName**
  - Object name

- **bPicUpUseTransColor**
  - Transparent color yes/no

Return value

- **TRUE**
  - The function has been completed without any errors.

- **FALSE**
  - An error has occurred.

**SetPicUseTransColor**

Function

Controls the transparent color of the background picture of a graphic object.

Syntax

```c
BOOL SetPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName,
BOOL bPicUseTransColor);
```

Parameters

- **lpszPictureName**
  - Picture name

- **lpszObjectName**
  - Object name

- **bPicUseTransColor**
  - Transparent color yes/no
ANSI-C for Creating Functions and Actions

2.15 ANSI-C function descriptions

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

property

Property - short description

The properties of objects for which there are no direct functions can be modified or called in using the functions in the Property group.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetPropBOOL

Function

Sets a property with the value "bValue".

Syntax

BOOL SetPropBOOL(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, BOOL bValue)

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lpszPropertyName
Name of the object property

bValue
Value in BOOL data format
Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the function is called for the picture object, set the parameter lpszObjectName = NULL.

See also
SetPropBOOL example (Page 1568)

SetPropChar

Function
Sets a property with the value the pointer "szValue" points to.

Syntax
BOOL SetPropChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, char* szValue)

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lpszPropertyName
Name of the object property

szValue
Pointer to the value

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
If the function is called for the picture object, set the parameter lpszObjectName = NULL.

See also
GetPropChar example (Page 1534)

SetPropDouble

Function
Sets a property with the value "dValue".

Syntax
BOOL SetPropDouble(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, double dValue)

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lpszPropertyName
Name of the object property

dValue
Value in "double" data format

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
If the function is called for the picture object, set the parameter lpszObjectName = NULL.

SetPropWord

Function
Sets a property with the value "IValue".

Syntax
BOOL SetPropWord(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, long lValue)

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lpszPropertyName
Name of the object property

lValue
Value in "long" data format

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

Note
If the function is called for the picture object, set the parameter lpszObjectName = NULL.
**State - short description**

The properties of status displays can be modified or called in using the functions in the State group.

---

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

---

### SetBasePicTransColor

**Function**

Sets the transparent color of the basic picture for the status display.

**Syntax**

```c
BOOL SetBasePicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBasePicTransColor);
```

**Parameters**

- **lpszPictureName**
  Picture name

- **lpszObjectName**
  Object name

- **lBasePicTransColor**
  Transparent color of the basic picture

**Return value**

**TRUE**

The function has been completed without any errors.
FALSE
An error has occurred.

Note
This function only applies to Bitmap graphics (*.bmp).

See also
Color chart (Page 1592)
SetBackColor example (Page 1559)

SetBasePicUseTransColor

Function
Controls the transparent color of the basic picture for the status display.

Syntax
BOOL SetBasePicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bBasePicUseTransColor);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

bBasePicUseTransColor
Transparent color yes/no

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
SetFlashFlashPicture

Function
Specifies whether the flash picture of the status display is animated dynamically or statically.

Syntax
BOOL SetFlashFlashPicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashFlashPicture);

Parameters
- **lpszPictureName**
  Picture name
- **lpszObjectName**
  Object name
- **bFlashFlashPicture**
  Type of flash picture
  - TRUE dynamically animated flash picture
  - FALSE statically animated flash picture

Return value
- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

SetFlashPicTransColor

Function
Sets the transparent color of the flash picture for a status display.

Syntax
BOOL SetFlashPicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashPicTransColor);
Parameters

**lpszPictureName**  
Picture name

**lpszObjectName**  
Object name

**lFlashPicTransColor**  
Transparent color of the flash picture

Return value

**TRUE**  
The function has been completed without any errors.

**FALSE**  
An error has occurred.

Note

This function only applies to Bitmap graphics (*.bmp).

See also

Color chart (Page 1592)
SetBackColor example (Page 1559)

**SetFlashPicUseTransColor**

Function

Controls the transparent color of the flash picture for a status display.

Syntax

```c
BOOL SetFlashPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashPicUseTransColor);
```

Parameters

**lpszPictureName**  
Picture name
**IpszObjectName**  
Object name

**bFlashPicUseTransColor**  
Transparent color yes/no

**Return value**

**TRUE**  
The function has been completed without any errors.

**FALSE**  
An error has occurred.

**SetFlashRateFlashPic**

**Function**  
Sets the flash frequency of the flash picture for a status display.

**Syntax**

```c
BOOL SetFlashRateFlashPic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateFlashPic);
```

**Parameters**

**IpszPictureName**  
Picture name

**IpszObjectName**  
Object name

**lFlashRateFlashPic**  
Flash frequency of the flash picture

**Return value**

**TRUE**  
The function has been completed without any errors.
FALSE
An error has occurred.

Note
Since the flashing is performed by means of software engineering, the precise frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time etc.).

See also
Flash frequencies (Page 1590)
SetFlashRateFlashPic example (Page 1563)

SetIndex

Function
Sets the index of a polygon or polyline thus defining the current object point.

Syntax
BOOL SetIndex(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lIndex);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

Index
Index value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
style

Style - short description
Various properties affecting the appearance of objects can be modified or called in using the functions in the Style group.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

SetBackBorderWidth

Function
Sets the frame width of 3D frames and slider objects.

Syntax
BOOL SetBackBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int IBackBorderWidth);

Parameters

lpzPictureName
Picture name

lpzObjectName
Object name

IBackBorderWidth
Frame width in pixels

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

SetBorderStyle example (Page 1560)

**SetBorderEndStyle**

**Function**

Sets the type of line end.

**Syntax**

```c
BOOL SetBorderEndStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderEndStyle);
```

**Parameters**

- `lpszPictureName` Picture name
- `lpszObjectName` Object name
- `lBorderEndStyle` Type of line end as numeric value

**Return value**

- **TRUE**
  The function has been completed without any errors.
- **FALSE**
  An error has occurred.

See also

- Line end style (Page 1595)
- SetBorderEndStyle example (Page 1559)

**SetBorderStyle**

**Function**

Sets the line or border style.
Syntax

BOOL SetBorderStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderStyle);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lBorderStyle
Numeric value defining the line or border style

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Line styles (Page 1595)
SetBorderStyle example (Page 1560)

SetBorderWidth

Function

Sets the line or border line width.

Syntax

BOOL SetBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderWidth);

Parameters

lpszPictureName
Picture name
IpszObjectName
Object name

IBorderWidth
Line width or border line width

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
SetBorderStyle example (Page 1560)

SetBoxAlignment

Function
Defines the arrangement of controls (left or right justified) in check boxes or radio boxes.

Syntax
BOOL SetBoxAlignment(LPCTSTR IpszPictureName, LPCTSTR IpszObjectName, long int IBoxAlignment);

Parameters

IpszPictureName
Picture name

IpszObjectName
Object name

IBoxAlignment
Arrangement of controls

Return value

TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

See also
Element alignment in check boxes and radio boxes (Page 1592)
SetBorderStyle example (Page 1560)

SetFillStyle

Function
Sets the type of fill pattern.

Syntax
BOOL SetFillStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillStyle);

Parameters
lpszPictureName
Picture name
lpszObjectName
Object name
IFillStyle
Type of fill pattern as numeric value

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

Note
If the function is called in relation to the entire picture, set the parameter lpszObjectName = ZERO.
See also

Fill pattern (Page 1594)
SetFillStyle example (Page 1562)

SetFillStyle2

Function

Sets the bar fill pattern for a bar graph.

Syntax

BOOL SetFillStyle2(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillStyle2);

Parameters

*lpszPictureName
Picture name

*lpszObjectName
Object name

*IFillStyle2
Bar fill pattern as numeric value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Fill pattern (Page 1594)
SetFillStyle example (Page 1562)
SetItemBorderStyle

Function

Sets the dividing line style for the "text list" object.

Syntax

BOOL SetItemBorderStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderStyle);

Parameters

lpszPictureName
Picture name

lpszObjectName
Object name

lItemBorderStyle
Numeric value defining the dividing line style

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Line styles (Page 1595)
SetBorderStyle example (Page 1560)

SetItemBorderWidth

Function

Sets the dividing line width for the "text list" object.

Syntax

BOOL SetItemBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderWidth);
Parameters

- `lpszPictureName`  
  Picture name

- `lpszObjectName`  
  Object name

- `lItemBorderWidth`  
  Numeric value defining the dividing line width

Return value

- `TRUE`  
  The function has been completed without any errors.

- `FALSE`  
  An error has occurred.

See also

SetBorderStyle example (Page 1560)

SetPressed

Function

Specifies for buttons or round buttons whether the switch setting is "pressed" or "not pressed".

Syntax

```c
BOOL SetPressed(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPressed);
```

Parameters

- `lpszPictureName`  
  Picture name

- `lpszObjectName`  
  Object name
bPressed
Switch setting of the button

    TRUE   Switch setting "pressed"
    FALSE  Switch setting "not pressed"

Return value

    TRUE
    The function has been completed without any errors.

    FALSE
    An error has occurred.

SetToggle
Function
Specifies for buttons or round buttons whether the switch is latchable or not.

Syntax

    BOOL SetToggle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bToggle);

Parameters

    lpszPictureName
    Picture name

    lpszObjectName
    Object name

    bToggle
    Switch latchable/not latchable

Return value

    TRUE
    The function has been completed without any errors.

    FALSE
    An error has occurred.
**SetWindowsStyle**

**Function**

Specifies whether buttons are to be displayed in Windows style.

**Syntax**

```c
BOOL SetWindowsStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bWindowStyle);
```

**Parameters**

- `lpszPictureName`  
  Picture name

- `lpszObjectName`  
  Object name

- `bWindowStyle`  
  "Windows style" on/off

**Return value**

- **TRUE**  
  The function has been completed without any errors.

- **FALSE**  
  An error has occurred.

---

**OpenHomePicture**

**Function**

Opens the entered start picture.

**Syntax**

```c
BOOL OpenHomePicture();
```

**Return value**

- **TRUE**  
  The function has been completed without any errors.
FALSE
An error has occurred.

OpenNextPicture

Function

WinCC saves the names of the pictures opened by the user during runtime as well as the sequence in which these pictures were opened.

The maximum number of picture names saved this way can be set in the WinCC Explorer in the computer properties on the "Graphics Runtime" tab under "picture buffer size".

The OpenNextPicture function now opens the picture which was opened before the last call of OpenPrevPicture.

Syntax

BOOL OpenNextPicture();

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

OpenPrevPicture

Function

WinCC saves the names of the pictures opened by the user during runtime as well as the sequence in which these pictures were opened.

The maximum number of picture names saved this way can be set in the WinCC Explorer in the computer properties on the "Graphics Runtime" tab under "picture buffer size".

The OpenPrevPicture function now opens the picture which was opened before the currently open picture.

Syntax

BOOL OpenPrevPicture();
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

OpenStoredPicture

**Function**
Opens the picture saved with the StorePicture function.

**Syntax**

```c
BOOL OpenStoredPicture();
```

**Return value**

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

StorePicture

**Function**
Saves the current picture which can then be opened with the OpenStoredPicture function.

**Syntax**

```c
BOOL StorePicture();
```

**Return value**

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.
2.15.3.5 tag

tag - short description

Tags can be set or called in using the functions from the tag group.

GetTag or GetTagWait?

Process tags that are called with GetTag are put in an image. Since updating and reading the image is done in two separate procedures, the GetTag call is not directly influenced by the coupling. It can therefore be executed quicker and more independently than a GetTagWait retrieval.

With GetTagWait, process tags that have been requested are not accepted in the image. A GetTagWait retrieval reads the value explicitly from the AS. This always includes the send and return path through the coupling and the response time of the AS. During this runtime, the processing of the C actions is blocked and the time required for the retrieval cannot be estimated. If multiple tags are read, the time is added.

A GetTagWait call is required if
  • fast write/read procedures are to be synchronized
  • a value is read explicitly from the AS
  • or a registration is to be avoided in the image deliberately.

The GetTagWait call is to be avoided in cyclic C-Actions, this is the main reason for performance problems.

SetTag or SetTagWait?

The SetTag retrieval distributes a write job without waiting for confirmation from the AS.

The SetTagWait retrieval distributes a write job and waits for confirmation from the AS. This always includes the send and return path through the coupling and the response time of the AS. During this runtime, the processing of the C actions is blocked and the time required for the retrieval cannot be estimated. If multiple tags are written, the time is added.

A SetTagWait call is set to guarantee that the value has been written before the C-Action is processed any further. The SetTagWait call in cyclic C actions is to be avoided.

Note

The difference between GetTag and GetTagWait also exists for internal tags. The difference is not quite so serious here however, since no coupling comes into play. To synchronize fast write/read procedures, the respective wait function is to be used with internal tags as well.

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.
get

Functionality of the GetTag functions

GetTagXXX

By calling the function the tag is logged on and, from that moment, polled cyclically from the AS. The cycle for the registration depends on the trigger (see following description). For GetTagXXX calls, the value that is available in WinCC is sent. For Close Picture, the tag actions are ended again.

The call is marked by the following:
- The value is read from the tag image by WinCC.
- The call is faster in comparison to GetTagXXXWait (except for the first call which generally takes longer because the value from the PLC must be read out and logged on).
- The duration of the call does not depend on the bus-load or on the AS.
- The function does not deliver any information on the status of the tags

Asynchronous

Note
If a tag is requested in a Global Script action, it remains registered throughout the enter Runtime of WinCC.

In Callback functions, the respective GetTagXXXWait function must be used.

Behavior in actions with tag trigger (recommended):
All of the tags contained in the tag trigger are already known with Open Picture and are registered with the defined monitoring time.

Since all tags are requested at once, the best possible optimization can be targeted from the channel. If a tag is requested with GetTagXXX() within a C-Action, which is contained in the trigger, the value already exists and is sent directly to the call (high-performance).

Registering tags in actions with tag trigger
As the tags are already known when the picture is selected, they can be transmitted in a job to the Data Manager and so be registered collectively to the channel.

Note
If a tag is requested, which is not in the trigger, then the behavior is the same as with the default trigger.

Behavior in actions with default trigger:
tag are registered with half of the cycle time with the first call. For every other call, the value is present.

Registering tags in actions with default trigger and event trigger
Only when the individual actions are executed is it identified which tags are needed in the picture. As a result, the tags are registered to the channel in a large number of single jobs. When a picture with cyclic actions is selected, the continual reorganization may place a heavy strain on communications.

Example: The channel supports custom cycle creation. Usually cycles are created by the channel directly from the AS. The resources for these cycles are limited by the AS. As a result, the channel stops the current jobs for this cycle and reconfigures the cycle on the AS.
Behavior in event triggered actions:
The tag is registered in the "upon change" mode with the first call. Process tags that are registered in the "upon change" mode correspond with a cyclic read job with a cycle time of 1s.

Note
If a value is requested by GetTagXXX() by a mouse click for example, the tag is accepted in the tag image. The tag is requested cyclically from the AS as of this point in time and therefore increases the basic load.

To avoid this increase in basic loading, the value can be requested by GetTagXXXWait(). The call GetTagXXXWait() causes a higher communication load one time but the tag is not added to the tag image.

GetTagXXXWait
The function returns the current value. The tag is not registered cyclically, the value is requested from the AS one time only.

The call is marked by the following:
• The value is read explicitly from the AS.
• The call, compared with GetTagXXX, takes longer.
• The duration of the call does not depend on the bus-load or on the AS.
• The function does not deliver any information on the status of the tags.

**Synchronous**

GetTagXXXState

The function GetTagXXXState has the same features as GetTagXXX, it also sends the function information on the status of the tags. Since the status is always delivered internally, there is no performance difference to GetTagXXX.

GetTagXXXStateWait

The function GetTagXXXStateWait has the same features as GetTagXXXWait, additionally it sends the function information on the status of the tags. Since the status is always delivered internally, there is no performance difference to GetTagXXXWait.

The difference between functions GetTagXXXStateWait and GetTagXXXState corresponds with the difference between GetTagXXXWait and GetTagXXX. Since the value is explicitly read from the AS for process tags, the value and the status can be more current than for GetTagXXXState.

GetTagXXXStateQC

The function GetTagXXXStateQC has the same features as GetTagXXXState. The function also delivers information on the quality code of the tag.

GetTagXXXStateQC Wait

The function GetTagXXXStateQCWait has the same features as GetTagXXXStateWait. The function also delivers information on the quality code of the tag.

GetTagMultiWait

The function GetTagMultiWait has the same features as GetTagXXXWait. However, it allows the request for more tags in a job. Therefore, the read requests in the direction of the AS can be optimized in most cases so that only one request will be given to the AS.
GetTagMultiStateWait

The function GetTagMultiStateWait has the same features as GetTagMultiWait, additionally it sends the function information on the statuses of the tags.

GetTagMultiStateQCWait

The function GetTagMultiStateQCWait has the same features as GetTagMultiStateWait. The function also delivers information on the quality codes of the tags.

getTagBitStateWait

Function

Determines the value of a tag of data type "Binary tag". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

```c
BOOL GetTagBitStateWait(Tag Tag_Name, PDWORD lp_dwstate);
```

Parameters

- **Tag_Name**
  
  name of the tag

- **lp_dwstate**
  
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "BOOL"

See also

- Tag statuses (Page 1598)
- GetTagBitStateWait example (Page 1541)
- Functionality of the GetTag functions (Page 1409)
GetTagByteStateWait

Function

Determines the value of a tag of data type "unsigned 8 bit". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

BYTE GetTagByteStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "BYTE"

See also

Tag statuses (Page 1598)
GetTagWordStateWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagCharStateWait

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

char* GetTagCharStateWait(Tag Tag_Name, PDWORD lp_dwstate);
Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Pointer to the value of the tag in data type "char".
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}

See also

Functionality of the GetTag functions (Page 1409)
Tag statuses (Page 1598)
Beispiel GetTagCharStateWait (Page 1544)

GetTagDoubleStateWait

Function

Determines the value of a tag of data type "64-bit floating point value". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

double GetTagDoubleStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
GetTagDWordStateWait

Function
Determines the value of a tag of data type "unsigned 32 bit". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax
DWORD GetTagDWordStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters
- **Tag_Name**
  name of the tag
- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
Value of the tag in the data type "DWORD"

See also
- Tag statuses (Page 1598)
- GetTagWordStateWait example (Page 1554)
- Functionality of the GetTag functions (Page 1409)
GetTagFloatStateWait

Function

Determines the value of a tag of data type "32-bit floating point value". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

float GetTagFloatStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

**Tag_Name**
name of the tag

**lp_dwstate**
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "float"

See also

Tag statuses (Page 1598)
GetTagFloatStateWait example (Page 1546)
Functionality of the GetTag functions (Page 1409)

GetTagMultiStateWait

Function

The values and states of several tags are established and stored in the corresponding addresses in the specified format. The values are read explicitly from the AS.

The function must transfer a DWORD array whose members contain the individual tag states after the function is invoked. The size of the array must be selected so that sufficient memory space is available for these statuses.

Syntax

BOOL GetTagMultiStateWait(DWORD* pdwState, const char* pFormat)
GetTagRawStateWait

Function
Determines the value of a tag of data type "Raw data type". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax
BOOL GetTagRawStateWait(Tag Tag_Name, BYTE pValue, DWORD size, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag
size
Size of the byte field in bytes

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
GetTagRawStateWait example (Page 1551)
Functionality of the GetTag functions (Page 1409)

GetTagSByteStateWait

Function
Determines the value of a tag of data type "signed 8 bit". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax
signed char GetTagSByteStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
The value of the tag in the data type "signed char"
GetTagSDWordStateWait

Function

Determines the value of a tag of data type "signed 32 bit". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

```c
long GetTagSDWordStateWait(Tag Tag_Name, PDWORD lp_dwstate);
```

Parameters

- **Tag_Name**
  name of the tag

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "long"

See also

- Tag statuses (Page 1598)
- GetTagSByteStateWait example (Page 1553)
- Functionality of the GetTag functions (Page 1409)

GetTagSWordStateWait

Function

Determines the value of a tag of data type "signed 16 bit". The value is read explicitly from the AS. The status of the tag is also returned.
Syntax

short GetTagSWordStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "short"

See also

Functionality of the GetTag functions (Page 1409)
Tag statuses (Page 1598)
GetTagSByteStateWait example (Page 1553)

GetTagWordStateWait

Function

Determines the value of a tag of data type "unsigned 16 bit". The value is read explicitly from the AS. The status of the tag is also returned.

Syntax

WORD GetTagWordStateWait(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
Return value

Value of the tag in the data type "WORD"

See also

Tag statuses (Page 1598)
GetTagWordStateWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagBitState

Function

Determines the value of a tag of data type "Binary tag". The status of the tag is also returned.

Syntax

BOOL GetTagBitState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "BOOL"

See also

Tag statuses (Page 1598)
GetTagBitStateWait example (Page 1541)
Functionality of the GetTag functions (Page 1409)

GetTagByteState

Function

Determines the value of a tag of data type "unsigned 8 bit". The status of the tag is also returned.
Syntax

BYTE GetTagByteState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "BYTE"

See also

Tag statuses (Page 1598)
GetTagWordStateWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagCharState

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag". The status of the tag is also returned.

Syntax

char* GetTagCharState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
Return value

Pointer to the value of the tag in data type "char".

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":
```
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......
}
```

See also

- Tag statuses (Page 1598)
- Beispiel GetTagCharStateWait (Page 1544)
- Functionality of the GetTag functions (Page 1409)

GetTagDoubleState

Function

Determines the value of a tag of data type "64-bit floating point value". The status of the tag is also returned.

Syntax

```
double GetTagDoubleState(Tag Tag_Name, PDWORD lp_dwstate);
```

Parameters

- **Tag_Name**
  name of the tag

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "double"
See also

Tag statuses (Page 1598)
GetTagFloatStateWait example (Page 1546)
Functionality of the GetTag functions (Page 1409)

GetTagDWordState

Function

Determines the value of a tag of data type "unsigned 32 bit". The status of the tag is also returned.

Syntax

DWORD GetTagDWordState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "DWORD"

See also

Tag statuses (Page 1598)
GetTagWordStateWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagFloatState

Function

Determines the value of a tag of data type "32-bit floating point value". The status of the tag is also returned.
Syntax

float GetTagFloatState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "float"

See also

GetTagFloatStateWait example (Page 1546)
Tag statuses (Page 1598)
Functionality of the GetTag functions (Page 1409)

GetTagRawState

Function

Determines the value of a tag of data type "Raw data type". The status of the tag is also returned.

Syntax

BOOL GetTagRawState(Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag

size
Size of the byte field in bytes
lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
GetTagRawStateWait example (Page 1551)
Functionality of the GetTag functions (Page 1409)

GetTagSByteState

Function
Determines the value of a tag of data type "signed 8 bit". The status of the tag is also returned.

Syntax
signed char GetTagSByteState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
The value of the tag in the data type "signed char"
See also
Tag statuses (Page 1598)
GetTagSByteStateWait example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagSDWordState

Function
Determines the value of a tag of data type "signed 32 bit". The status of the tag is also returned.

Syntax
long GetTagSDWordState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
Value of the tag in the data type "long"

See also
Tag statuses (Page 1598)
GetTagSByteStateWait example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagSWordState

Function
Determines the value of a tag of data type "signed 16 bit". The status of the tag is also returned.

Syntax
short GetTagSWordState(Tag Tag_Name, PDWORD lp_dwstate);
 Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "short"

See also

Tag statuses (Page 1598)
GetTagSByteStateWait example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagWordState

Function

Determines the value of a tag of data type "unsigned 16 bit". The status of the tag is also returned.

Syntax

WORD GetTagWordState(Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

Value of the tag in the data type "WORD"
GetTagBitStateQCWait

Function

Determines the value of a tag of data type "Binary tag". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

BOOL GetTagBitStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tags in the data type "BOOL".

See also

Tag statuses (Page 1598)
GetTagWordStateWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)
GetTagByteStateQCWait

Function

Determines the value of a tag of data type "unsigned 8 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

BYTE GetTagByteStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "BYTE".

See also

Tag statuses (Page 1598)
GetTagWordStateQCWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagCharStateQCWait

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

char* GetTagCharStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
Parameters

**Tag_Name**
Name of the tag.

**lp_dwstate**
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

**pdwQualityCode**
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Pointer to the value of the tag in data type "char".

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()";

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    .......... 
}
```

See also

- Tag statuses (Page 1598)
- GetTagCharStateQCWait example (Page 1543)
- Functionality of the GetTag functions (Page 1409)

**GetTagDoubleStateQCWait**

Function

Determines the value of a tag of data type "64-bit floating point value". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

```c
double GetTagDoubleStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

Parameters

**Tag_Name**
Name of the tag.
GetTagDWordStateQCWait

Function
Determine the value of a tag of data type "unsigned 32 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax
DWORD GetTagDWordStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters
Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value
Value of the tag in the data type "DWORD".
See also

Tag statuses (Page 1598)
GetTagWordStateQCWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagFloatStateQCWait

Function

Determines the value of a tag of data type "32-bit floating point value". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

float GetTagFloatStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "float".

See also

Tag statuses (Page 1598)
GetTagFloatStateQCWait example (Page 1545)
Functionality of the GetTag functions (Page 1409)
GetTagMultiStateQCWait

Function

The values, states and quality codes are determined for several tags and are stored in the respective addresses in the specified format. The values are read explicitly from the AS.

The function must be provided with two DWORD arrays, the member of which contains the states and quality codes of the individual tags after the function has been called. The size of the arrays must be selected so that sufficient memory space is available for these statuses.

Syntax

BOOL GetTagMultiStateQCWait(DWORD* pdwState, DWORD* pdwQualityCode, const char* pFormat)

Parameters

pdwState
Field in which the status of the individual tags is stored after the function has been completed.

pdwQualityCode
Field in which the quality codes of the individual tags is stored after the function has been completed.

pFormat
Format description for all requested tags and for each tag name and address of the value.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Format descriptors (Page 1593)
Tag statuses (Page 1598)
GetTagMultiStateQCWait example (Page 1547)
Functionality of the GetTag functions (Page 1409)
GetTagRawStateQCWait

Function

Determines the value of a tag of data type "Raw data type". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

BOOL GetTagRawStateQCWait(Tag Tag_Name, BYTE pValue, DWORD size, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

pValue
Pointer to a byte field containing the value of the raw data tag.

size
Size of the byte field in bytes.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
GetTagRawStateQCWait example (Page 1550)
Functionality of the GetTag functions (Page 1409)
GetTagSByteStateQCWait

Function

Determines the value of a tag of data type "signed 8 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

signed char GetTagSByteStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "signed char".

See also

Tag statuses (Page 1598)
GetTagSByteStateQCWait example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSDWordStateQCWait

Function

Determines the value of a tag of data type "signed 32 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.
Syntax

long GetTagSDWordStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tags in the data type "long".

See also

Tag statuses (Page 1598)
GetTagSByteStateQCWait example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSWordStateQCWait

Function

Determines the value of a tag of data type "signed 16 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

short GetTagSWordStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.
Ip_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value
Value of the tag in the data type "short".

See also
Tag statuses (Page 1598)
GetTagSByteStateQCWait example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagValueStateQCWait

Function
Enables the transfer of a value in the form of a variant. Establishes the pointer to the result structure containing the value. The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax
BOOL GetTagValueStateQCWait(LPDM_VARKEY IpdmVarKey,
LPDM_VAR_UPDATE_STRUCTEX Ipdmresult, LPCMN_ERROR IpdmError);

Parameters
IpdmVarKey
Pointer to a structure of the data type "DM_VARKEY"

Ipdmresult
Pointer to the value from data type "DM_VAR_UPDATE_STRUCTEX"

IpdmError
Pointer to the structure which contains the error description
Return value

**TRUE**
The function has been completed without any errors.

**FALSE**
An error has occurred.

See also

- Structure definition CMN_ERROR (Page 1601)
- Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1604)
- Structure definition DM_VARKEY (Page 1605)
- Functionality of the GetTag functions (Page 1409)

GetTagWordStateQCWait

Function

Determines the value of a tag of data type "unsigned 16 bit". The value is read explicitly from the AS. In addition, the status and the quality code of the tags are returned.

Syntax

```
WORD GetTagWordStateQCWait(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

Parameters

**Tag_Name**
Name of the tag.

**lp_dwstate**
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

**pdwQualityCode**
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "WORD".
GetTagBitStateQC

Function
Determines the value of a tag of data type "Binary tag". In addition, the status and the quality code of the tags are returned.

Syntax
BOOL GetTagBitStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value
Value of the tags in the data type "BOOL".

See also
Tag statuses (Page 1598)
GetTagBitStateQC example (Page 1540)
Functionality of the GetTag functions (Page 1409)
GetTagByteStateQC

Function

Determines the value of a tag of data type "unsigned 8 bit". In addition, the status and the quality code of the tags are returned.

Syntax

BYTE GetTagByteStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "BYTE".

See also

Tag statuses (Page 1598)
GetTagWordStateQCWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

GetTagCharStateQC

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag". In addition, the status and the quality code of the tags are returned.
Syntax

```c
char* GetTagCharStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

Parameters

- **Tag_Name**
  Name of the tag.

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

- **pdwQualityCode**
  Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Pointer to the value of the tag in data type "char".

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}
```

See also

- Tag statuses (Page 1598)
- GetTagCharStateQCWait example (Page 1543)
- Functionality of the GetTag functions (Page 1409)

**GetTagDoubleStateQC**

Function

Determines the value of a tag of data type "64-bit floating point value". In addition, the status and the quality code of the tags are returned.

Syntax

```c
double GetTagDoubleStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```
Parameters

**Tag_Name**
Name of the tag.

**lp_dwstate**
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

**pdwQualityCode**
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "double".

See also

GetTagFloatStateQCWait example (Page 1545)
Tag statuses (Page 1598)
Functionality of the GetTag functions (Page 1409)

GetTagDWordStateQC

Function

Determines the value of a tag of data type "unsigned 32 bit". In addition, the status and the quality code of the tags are returned.

Syntax

```
DWORD GetTagDWordStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

Parameters

**Tag_Name**
Name of the tag.

**lp_dwstate**
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
GetTagFloatStateQC

Function

Determines the value of a tag of data type "32-bit floating point value". In addition, the status and the quality code of the tags are returned.

Syntax

float GetTagFloatStateQC(Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "float".
GetTagRawStateQC

Function

Determines the value of a tag of data type "Raw data type". In addition, the status and the quality code of the tags are returned.

Syntax

BOOL GetTagRawStateQC(Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

pValue
Pointer to a byte field containing the value of the raw data tag.

size
Size of the byte field in bytes.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
GetTagSByteStateQC

Function

Determines the value of a tag of data type "signed 8 bit". In addition, the status and the quality code of the tags are returned.

Syntax

```c
signed char GetTagSByteStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

Parameters

- **Tag_Name**
  Name of the tag.

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

- **pdwQualityCode**
  Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tag in the data type "signed char".

See also

- Tag statuses (Page 1598)
- GetTagRawStateQCWait example (Page 1550)
- Functionality of the GetTag functions (Page 1409)
GetTagSDWordStateQC

Function

Determines the value of a tag of data type "signed 32 bit". In addition, the status and the quality code of the tags are returned.

Syntax

long GetTagSDWordStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

Return value

Value of the tags in the data type "long".

See also

Tag statuses (Page 1598)
GetTagSByteStateQCWait example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSWordStateQC

Function

Determines the value of a tag of data type "signed 16 bit". In addition, the status and the quality code of the tags are returned.
**Syntax**

```c
short GetTagSWordStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);
```

**Parameters**

- **Tag_Name**  
  Name of the tag.

- **lp_dwstate**  
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

- **pdwQualityCode**  
  Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.

**Return value**

Value of the tag in the data type "short".

**See also**

- Tag statuses (Page 1598)
- GetTagSByteStateQCWait example (Page 1552)
- Functionality of the GetTag functions (Page 1409)

**GetTagValueStateQC**

**Function**

Enables the transfer of a value in the form of a variant. Establishes the pointer to the result structure containing the value. In addition, the status and the quality code of the tags are returned.

**Syntax**

```c
BOOL GetTagValueStateQC(LPDM_VARKEY lpdmVarKey,
                        LPDM_VAR_UPDATE_STRUCTEX lpdmresult, LPCMN_ERROR lpdmError);
```

**Parameters**

- **lpdmVarKey**  
  Pointer to a structure of the data type "DM_VARKEY"
lpdmresult
Pointer to the value from data type "DM_VAR_UPDATE_STRUCTEX"

lpdmError
Pointer to the structure which contains the error description

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
Structure definition CMN_ERROR (Page 1601)
Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1604)
Structure definition DM_VARKEY (Page 1605)
Functionality of the GetTag functions (Page 1409)

GetTagWordStateQC

Function
Determines the value of a tag of data type "unsigned 16 bit". In addition, the status and the quality code of the tags are returned.

Syntax
WORD GetTagWordStateQC(Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name
Name of the tag.

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

pdwQualityCode
Pointer to a DWORD in which the quality code of the tag is stored after the function is complete.
Return value

Value of the tag in the data type "WORD".

See also

Tag statuses (Page 1598)
GetTagWordStateQCWait example (Page 1554)
Functionality of the GetTag functions (Page 1409)

wait

GetTagBitWait

Function

 Determines the value of a tag of data type "Binary tag". The value is read explicitly from the AS.

Syntax

BOOL GetTagBitWait(Tag Tag_Name);

Parameters

Tag_Name

name of the tag

Return value

Value of the tag in the data type "BOOL"

See also

GetTagBlt example (Page 1539)
Functionality of the GetTag functions (Page 1409)
GetTagByteWait

Function
Determines the value of a tag of data type "unsigned 8 bit". The value is read explicitly from the AS.

Syntax
BYTE GetTagByteWait(Tag Tag_Name);

Parameters
Tag_Name
name of the tag

Return value
Value of the tag in the data type "BYTE"

See also
GetTagWord example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagCharWait

Function
Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag". The value is read explicitly from the AS.

Syntax
char* GetTagCharWait(Tag Tag_Name);

Parameters
Tag_Name
name of the tag

Return value
Pointer to a character string containing the value of the tag.
The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

```c
pszValue = GetText(lpszPictureName, "Text1");
if (pszValue != NULL) {
    .......
}
```

See also

- GetTagChar example (Page 1542)
- Functionality of the GetTag functions (Page 1409)

**GetTagDoubleWait**

**Function**

Determines the value of a tag of data type "64-bit floating point value". The value is read explicitly from the AS.

**Syntax**

```c
double GetTagDoubleWait(Tag Tag_Name);
```

**Parameters**

- **Tag_Name**
  name of the tag

**Return value**

Value of the tag in the data type "double"

See also

- GetTagFloat example (Page 1544)
- Functionality of the GetTag functions (Page 1409)

**GetTagDWordWait**

**Function**

Determines the value of a tag of data type "unsigned 32 bit". The value is read explicitly from the AS.
### GetTagDWordWait

**Syntax**

```c
DWORD GetTagDWordWait(Tag Tag_Name);
```

**Parameters**

- **Tag_Name**
  - name of the tag

**Return value**

Value of the tag in the data type "DWORD"

**See also**

- Functionality of the GetTag functions (Page 1409)
- GetTagWord example (Page 1553)

### GetTagFloatWait

**Function**

Determines the value of a tag of data type "32-bit floating point value". The value is read explicitly from the AS.

**Syntax**

```c
float GetTagFloatWait(Tag Tag_Name);
```

**Parameters**

- **Tag_Name**
  - name of the tag

**Return value**

Value of the tag in the data type "float"

**See also**

- GetTagFloat example (Page 1544)
- Functionality of the GetTag functions (Page 1409)
GetTagMultiWait

Function

The values of several tags are established and stored in the corresponding addresses in the specified format. The value is read explicitly from the AS. The memory for the tag value is created by the function with SysMalloc.

Syntax

BOOL GetTagMultiWait(const char* pFormat,...)

Parameters

pFormat
Format description for all requested tags and for each tag name and address of the value

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Format descriptors (Page 1593)
GetTagMultiWait example (Page 1549)
Functionality of the GetTag functions (Page 1409)

GetTagRawWait

Function

Determines the value of a tag of data type "Raw data type". The value is read explicitly from the AS.

Syntax

BOOL GetTagRawWait(Tag Tag_Name , BYTE pValue, DWORD size);
Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag

size
Size of the byte field in bytes

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Functionality of the GetTag functions (Page 1409)
GetTagRaw example (Page 1550)

GetTagSByteWait

Function
Determines the value of a tag of data type "signed 8 bit". The value is read explicitly from the AS.

Syntax

signed char GetTagSByteWait(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value
The value of the tag in the data type "signed char"
See also

GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSDWordWait

Function
Determines the value of a tag of data type "signed 32 bit". The value is read explicitly from the AS.

Syntax
long GetTagSDWordWait(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value
Value of the tag in the data type "long"

See also
GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSWordWait

Function
Determines the value of a tag of data type "signed 16 bit". The value is read explicitly from the AS.

Syntax
short GetTagSWordWait(Tag Tag_Name);
Parameters

Tag_Name
name of the tag

Return value

Value of the tag in the data type "short"

See also

GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagValueWait

Function

Enables the transfer of a value in the form of a variant. Establishes the pointer to the result structure containing the value. The value is read explicitly from the AS.

Syntax

BOOL GetTagValueWait(LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT lpdmresult, LPCMN_ERROR lpdmError);

Parameters

lpdmVarKey
Pointer to a structure of the data type "DM_VARKEY"

lpdmresult
Pointer to the value from data type "DM_VAR_UPDATE_STRUCT"

lpdmError
Pointer to the structure which contains the error description

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.
See also

- Structure definition CMN_ERROR (Page 1601)
- Structure definition DM_VAR_UPDATE_STRUCT (Page 1603)
- Structure definition DM_VARKEY (Page 1605)
- Functionality of the GetTag functions (Page 1409)

GetTagWordWait

Function

Determines the value of a tag of data type "unsigned 16 bit". The value is read explicitly from the AS.

Syntax

```c
WORD GetTagWordWait(Tag Tag_Name);
```

Parameters

- **Tag_Name**
  name of the tag

Return value

Value of the tag in the data type "WORD"

See also

- GetTagWord example (Page 1553)
- Functionality of the GetTag functions (Page 1409)

GetTagBit

Function

Determines the value of a tag of data type "Binary tag".

Syntax

```c
BOOL GetTagBit(Tag Tag_Name);
```
Parameters

**Tag_Name**
name of the tag

Return value

Value of the tag in the data type "BOOL"

See also

GetTagBit example (Page 1539)
Functionality of the GetTag functions (Page 1409)

GetTagByte

Function

Determines the value of a tag of data type "unsigned 8 bit".

Syntax

BYTE GetTagByte(Tag Tag_Name);

Parameters

**Tag_Name**
name of the tag

Return value

Value of the tag in the data type "BYTE"

See also

GetTagWord example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagChar

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag".
Syntax

cchar* GetTagChar(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value

Pointer to a character string containing the value of the tag.

The return value must be checked for validity to prevent a null pointer exception, e.g. with the function "GetText()":

pszValue = GetText(lpszPictureName,"Text1");
if(pszValue != NULL)
{
    ........
}

See also

GetTagChar example (Page 1542)
Functionality of the GetTag functions (Page 1409)

GetTagDateTime

Function

Determines the value of a tag of data type "Date/Time".

Syntax

SYSTEMTIME GetTagDateTime(Tag Tag_Name);

Parameter

Tag_Name
Name of the tag

Return value

Value of the tag in the data type "Date/Time".
GetTagDouble

Function

Determines the value of a tag of data type "64-bit floating point value".

Syntax

double GetTagDouble(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value

Value of the tag in the data type "double"

See also

GetTagFloat example (Page 1544)
Functionality of the GetTag functions (Page 1409)

GetTagDWord

Function

Determines the value of a tag of data type "unsigned 32 bit".

Syntax

DWORD GetTagDWord(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value

Value of the tag in the data type "DWORD"
See also

GetTagWord example (Page 1553)
Functionality of the GetTag functions (Page 1409)

GetTagFloat

Function

Determines the value of a tag of data type "32-bit floating point value".

Syntax

float GetTagFloat(Tag Tag_Name);

Parameters

Tag_Name
name of the tag

Return value

Value of the tag in the data type "float".

See also

GetTagFloat example (Page 1544)
Functionality of the GetTag functions (Page 1409)

GetTagRaw

Function

Determines the value of a tag of data type "Raw data type".

Syntax

BOOL GetTagRaw(Tag Tag_Name, BYTE* pValue, DWORD size);

Parameters

Tag_Name
name of the tag
GetTagSByte

Function

Determines the value of a tag of data type "signed 8 bit".

Syntax

`signed char GetTagSByte(Tag Tag_Name);`

Parameters

Tag_Name
name of the tag

Return value

The value of the tag in the data type "signed char"

See also

GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)
GetTagSDWord

Function
Determines the value of a tag of data type "signed 32 bit".

Syntax
long GetTagSDWord(Tag Tag_Name);

Parameters
Tag_Name
name of the tag

Return value
Value of the tag in the data type "long"

See also
GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetTagSWord

Function
Determines the value of a tag of data type "signed 16 bit".

Syntax
short GetTagSWord(Tag Tag_Name);

Parameters
Tag_Name
name of the tag

Return value
Value of the tag in the data type "short"
See also

GetTagSByte example (Page 1552)
Functionality of the GetTag functions (Page 1409)

GetValue

Function

Enables the transfer of a value in the form of a variant. Establishes the pointer to the result structure containing the value.

Syntax

BOOL GetTagValue(LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT lpdmresult, LPCMN_ERROR lpdmError);

Parameters

lpdmVarKey
Pointer to a structure of the data type "DM_VARKEY"

lpdmresult
Pointer to the value from data type "DM_VAR_UPDATE_STRUCT"

lpdmError
Pointer to the structure which contains the error description

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Structure definition CMN_ERROR (Page 1601)
Structure definition DM_VAR_UPDATE_STRUCT (Page 1603)
Structure definition DM_VARKEY (Page 1605)
Functionality of the GetTag functions (Page 1409)
GetTagWord

Function

Determines the value of a tag of data type "unsigned 16 bit".

Syntax

WORD GetTagWord(Tag Tag_Name);

Parameters

Tag_Name

name of the tag

Return value

Value of the tag in the data type "WORD"

See also

GetTagWord example (Page 1553)
Functionality of the GetTag functions (Page 1409)

set

Principle of the SetTag functions

SetTagXXX

The SetTagXXX function assigns the job a value to write and returns immediately to the caller. In this case, the system does not wait until value is actually written.

The call is marked by the following:

- The call is fast.
- The caller does not know when the value is actually written.
- The function provides no information on the state of the write job.

SetTagXXXWait

The function SetTagXXXWait assigns the job of writing a value and will first return to the caller when the value has actually been written.
The call is marked by the following:

- The call takes longer in comparison to SetTagXXX. The duration is also dependent on the channel and AS, amongst other things.
- The value is written after the call.
- The function provides no information on the state of the write job.

SetTagXXXState

The function SetTagXXXState has the same features as SetTagXXX; plus the function returns information regarding the status of the write request.

Since the status is always provided internally, there is no performance difference compared to SetTagXXX.

SetTagXXXStateWait

The function SetTagXXXStateWait has the same features as SetTagXXXWait; plus the function returns information regarding the status of the write request.

Since the status is always provided internally, there is no performance difference compared to SetTagXXXWait.

The difference between the functions SetTagXXXStateWait and SetTagXXXState corresponds to the difference between SetTagXXXWait and SetTagXXX.

Note, that certain statuses can only be generated when the write process has been completed.

SetTagMultiWait

The SetTagMultiWait function has the same features as SetTagXXXWait. It also offers the option of granting several write jobs in a single job.

state

wait

SetTagBitStateWait

Function

Sets the value of a tag of data type “Binary tag”. The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions

EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130
Syntax

```c
BOOL SetTagBitStateWait(Tag Tag_Name, short value, PDWORD lp_dwstate);
```

Parameters

- **Tag_Name**
  name of the tag

- **value**
  Value of the tag in the data type "short"

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

- **TRUE**
  The function itself has been completed without any errors.
  However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

- **FALSE**
  An error has occurred.

See also

- Tag statuses (Page 1598)
- Beispiel SetTagBitStateWait (Page 1570)

**SetTagByteStateWait**

Function

Sets the value of a tag of the data type "unsigned 8 Bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Syntax

```c
BOOL SetTagByteStateWait(Tag Tag_Name, BYTE value, PDWORD lp_dwstate);
```
Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "BYTE"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)

SetTagCharStateWait

Function

Sets the value of a tag of the data type "8-bit text tag" or "16-bit text tag". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions
EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagCharStateWait(Tag Tag_Name, LPSTR value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag
value
Value of the tag in the data type "LPSTR"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
SetTagCharStateWait example (Page 1571)

SetTagDoubleStateWait

Function
 Defines the value of a tag of the data type "64-bit floating point value". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagDoubleStateWait(Tag Tag_Name, double value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "double"
Ip_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
SetTagFloatStateWait example (Page 1572)

SetTagDWordStateWait

Function
Sets the value of a tag of the data type "unsigned 32 Bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions
EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagDWordStateWait(Tag Tag_Name, DWORD value, PDWORD Ip_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "DWORD"

Ip_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
Return value

**TRUE**
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

**FALSE**
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)

SetTagFloatStateWait

**Function**

Defines the value of a tag of the data type "32-bit floating point value". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions

EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

**Syntax**

```c
BOOL SetTagFloatStateWait(Tag Tag_Name, float value, PDWORD lp_dwstate);
```

**Parameters**

- **Tag_Name**
  name of the tag

- **value**
  Value of the tag in the data type "float"

- **lp_dwstate**
  Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

**Return value**

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

**FALSE**
An error has occurred.

**See also**
Tag statuses (Page 1598)
SetTagFloatStateWait example (Page 1572)

**SetTagMultiStateWait**

**Function**
Sets the values of several tags. The function is ended after the AS has acknowledged acceptance of the value.

The function must transfer a DWORD array whose members contain the individual tag states after the function is invoked. The size of the array must be selected so that sufficient memory space is available for these statuses.

**Syntax**

```c
BOOL SetTagMultiStateWait(DWORD* pdwState, const char* pFormat,...)
```

**Parameters**

**pdwState**
Field in which the tag statuses are stored.

**pFormat**
Format description for all requested tags and for each tag name and value.

**Return value**

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

**FALSE**
An error has occurred.
See also

Tag statuses (Page 1598)
SetTagMultiStateWait example (Page 1573)

SetTagRawStateWait

Function

Sets the value of a tag of the data type "Raw data type". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions

EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagRawStateWait(Tag Tag_Name, BYTE pValue, DWORD size, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag

size
Size of the byte field in bytes

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.
See also

Tag statuses (Page 1598)
SetTagRawStateWait example (Page 1575)

SetTagSByteStateWait

Function

Sets the value of a tag of the data type "signed 8 bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions

EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagSByteStateWait(Tag Tag_Name, signed char value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
The value of the tag in the data type "signed char"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
SetTagSDWordStateWait

Function

Sets the value of a tag of the data type "signed 32 bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions
EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagSDWordStateWait(Tag Tag_Name, long value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

Value
Value of the tag in the data type "long"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
SetTagSWordStateWait

Function

Sets the value of a tag of the data type "signed 16 bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions

EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagSWordStateWait(Tag Tag_Name, short value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "short"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
SetTagWordStateWait

Function
Sets the value of a tag of the data type "unsigned 16 Bit". The function is ended after the AS has acknowledged acceptance of the value. The status of the tag is also returned.

Principle of the SetTag functions


EXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagWordStateWait(Tag Tag_Name, WORD value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "WORD"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. To do this, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)
SetTagBitState

Function
Sets the value of a tag of data type "Binary tag". The status of the tag is also returned.

Syntax
BOOL SetTagBitState(Tag Tag_Name, short int value, PDWORD lp_dwstate);

Parameters
Tag_Name
name of the tag

value
Value of the tag in the data type "short int"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
Beispiel SetTagBitStateWait (Page 1570)
Principle of the SetTag functions (Page 1467)

SetTagByteState

Function
Sets the value of a tag of the data type "unsigned 8 Bit". The status of the tag is also returned.
Syntax

BOOL SetTagByteState(Tag Tag_Name, BYTE value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "BYTE"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagCharState

Function

Sets the value of a tag of the data type "8-bit text tag" or "16-bit text tag". The status of the tag is also returned.

Syntax

BOOL SetTagCharState(Tag Tag_Name, LPSTR value, PDWORD lp_dwstate);
Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "LPSTR"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
SetTagCharStateWait example (Page 1571)
Principle of the SetTag functions (Page 1467)

SetTagDoubleState

Function
Defines the value of a tag of the data type "64-bit floating point value". The status of the tag is also returned.

Syntax
BOOL SetTagDoubleState(Tag Tag_Name, double value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag
value
Value of the tag in the data type "double"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value
TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
SetTagFloatStateWait example (Page 1572)
Principle of the SetTag functions (Page 1467)

SetTagDWordState

Function
Sets the value of a tag of the data type "unsigned 32 Bit". The status of the tag is also returned.

Syntax
BOOL SetTagDWordState(Tag Tag_Name, DWORD value, PDWORD lp_dwstate);

Parameters
Tag_Name
name of the tag

value
Value of the tag in the data type "DWORD"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.
Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagFloatState

Function

Defines the value of a tag of the data type "32-bit floating point value". The status of the tag is also returned.

Syntax

BOOL SetTagFloatState(Tag Tag_Name, float value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "float"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also
Tag statuses (Page 1598)
SetTagFloatStateWait example (Page 1572)
Principle of the SetTag functions (Page 1467)

SetTagRawState

Function
Sets the value of a tag of the data type "Raw data type". The status of the tag is also returned.

Syntax
BOOL SetTagRawState(Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag

size
Size of the byte field in bytes

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors.
SetTagSByteState

Function

Sets the value of a tag of the data type "signed 8 bit". The status of the tag is also returned.

Syntax

BOOL SetTagSByteState(Tag Tag_Name, signed char value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
The value of the tag in the data type "signed char"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.

However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.
See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagSDWordState

Function

Sets the value of a tag of the data type "signed 32 bit". The status of the tag is also returned.

Syntax

BOOL SetTagSDWordState(Tag Tag_Name, long value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "long"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
Principle of the SetTag functions (Page 1467)
SetTagSWordState

Function

Sets the value of a tag of the data type "signed 16 bit". The status of the tag is also returned.

Syntax

BOOL SetTagSWordState(Tag Tag_Name, short value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "short"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagSByteStateWait (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagWordState

Function

Sets the value of a tag of the data type "unsigned 16 Bit". The status of the tag is also returned.
Syntax

BOOL SetTagWordState(Tag Tag_Name, WORD value, PDWORD lp_dwstate);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "short"

lp_dwstate
Pointer to a DWORD in which the status of the tag is stored after the function has been completed.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors. For this purpose, the tag status must be evaluated.

FALSE
An error has occurred.

See also

Tag statuses (Page 1598)
Beispiel SetTagWordStateWait (Page 1577)
Principle of the SetTag functions (Page 1467)

wait

SetTagBitWait

Function

Sets the value of a tag of data type "Binary tag". The function is ended after the AS has acknowledged acceptance of the value.

Syntax

BOOL SetTagBitWait(Tag Tag_Name, short value);
Parameter

Tag_Name
Name of the tag

Value
Value of the tag in the data type "short"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagBit example (Page 1570)
Principle of the SetTag functions (Page 1467)

SetTagByteWait

Function
Sets the value of a tag of the data type "unsigned 8 Bit". The function is ended after the AS has acknowledged acceptance of the value.

Syntax

BOOL SetTagByteWait(Tag Tag_Name, BYTE value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "BYTE"
Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagWord example (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagCharWait

Function
Sets the value of a tag of the data type "8-bit text tag" or "16-bit text tag". The function is ended after the AS has acknowledged acceptance of the value.

Syntax
BOOL SetTagCharWait(Tag Tag_Name, LPSTR value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "LPSTR"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.
See also

- Principle of the SetTag functions (Page 1467)
- SetTagChar example (Page 1570)

**SetTagDoubleWait**

**Function**

Defines the value of a tag of the data type "64-bit floating point value". The function is ended after the AS has acknowledged acceptance of the value.

**Syntax**

```c
BOOL SetTagDoubleWait(Tag Tag_Name, double value);
```

**Parameters**

- **Tag_Name**
  name of the tag

- **value**
  Value of the tag in the data type "double"

**Return value**

- **TRUE**
  The function itself has been completed without any errors.
  However, no test is made as to whether the tag could be written without errors.

- **FALSE**
  An error has occurred.

See also

- SetTagFloat example (Page 1571)
- Principle of the SetTag functions (Page 1467)

**SetTagDWordWait**

**Function**

Sets the value of a tag of the data type "unsigned 32 Bit". The function is ended after the AS has acknowledged acceptance of the value.
Syntax

BOOL SetTagDWordWait(Tag Tag_Name, DWORD value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "DWORD"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagWord example (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagFloatWait

Function

Defines the value of a tag of the data type "32-bit floating point value". The function is ended
after the AS has acknowledged acceptance of the value.

Syntax

BOOL SetTagFloatWait(Tag Tag_Name, float value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "float"
Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagFloat example (Page 1571)
Principle of the SetTag functions (Page 1467)

SetTagMultiWait

Function
The values of several tags are set in the specified format. The function is ended after the AS has acknowledged acceptance of the value.

Syntax
BOOL SetTagMultiWait(const char* pFormat, ...)

Parameters
pFormat
Format description for all requested tags and for each tag name and value.

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.
See also

Format descriptors (Page 1593)
SetTagMultiWait example (Page 1574)
Principle of the SetTag functions (Page 1467)

SetTagRawWait

Function

Sets the value of a tag of the data type "Raw data type". The function is ended after the AS has acknowledged acceptance of the value.

Syntax

BOOL SetTagRawWait(Tag Tag_Name, BYTE pValue, DWORD size);

Parameters

Tag_Name
name of the tag

pValue
The pointer to a byte field which contains the value of the raw data tag

size
Size of the byte field in bytes

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagRaw example (Page 1574)
Principle of the SetTag functions (Page 1467)
SetTagSByteWait

Function
Sets the value of a tag of the data type "signed 8 bit". The function is ended after the AS has acknowledged acceptance of the value.

Syntax
BOOL SetTagSByteWait(Tag Tag_Name, signed char value);

Parameters
Tag_Name
name of the tag

value
The value of the tag in the data type "signed char"

Return value
TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagSByte example (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagSDWordWait

Function
Sets the value of a tag of the data type "signed 32 bit". The function is ended after the AS has acknowledged acceptance of the value.

Syntax
BOOL SetTagSDWordWait(Tag Tag_Name, long value);
Parameters

**Tag_Name**
name of the tag

**value**
Value of the tag in the data type "long"

Return value

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

**FALSE**
An error has occurred.

See also

- Principle of the SetTag functions (Page 1467)
- SetTagSByte example (Page 1576)

**SetTagSWordWait**

Function

Sets the value of a tag of the data type "signed 16 bit". The function is ended after the AS has acknowledged acceptance of the value.

Syntax

```c
BOOL SetTagSWordWait(Tag Tag_Name, short value);
```

Parameters

**Tag_Name**
name of the tag

**value**
Value of the tag in the data type "short"
Return value

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

**FALSE**
An error has occurred.

See also

SetTagSByte example (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagValueWait

Function

Enables the transfer of a value in the form of a variant and sets the pointer to the value of the data type "Variant". The function is ended after the AS has acknowledged acceptance of the value.

Syntax

```c
BOOL SetTagValueWait(LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError);
```

Parameters

- **lpdmVarKey**
  Pointer to a structure of the data type "DM_VARKEY"

- **lpdmValue**
  Pointer to the value of data type "Variant". A description of the data type VARIANT can be found in the associated documentation.

- **dwState**
  Tag status which is returned after the function has been run.

- **lpdmError**
  Pointer to the structure which contains the error description
Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
Structure definition CMN_ERROR (Page 1601)
Tag statuses (Page 1598)
Structure definition DM_VAR_UPDATE_STRUCT (Page 1603)
Structure definition DM_VARKEY (Page 1605)
Principle of the SetTag functions (Page 1467)

SetTagWordWait

Function
Sets the value of a tag of the data type "unsigned 16 Bit". The function is ended after the AS has acknowledged acceptance of the value.

Syntax
BOOL SetTagWordWait(Tag Tag_Name, WORD value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "WORD"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.
FALSE
An error has occurred.

See also
SetTagWord example (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagBit

Function
Sets the value of a tag of data type "Binary tag".

Syntax
BOOL SetTagBit(Tag Tag_Name, short int value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "short int"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
Principle of the SetTag functions (Page 1467)
SetTagBit example (Page 1570)
SetTagByte

Function
Sets the value of a tag of the data type "unsigned 8 Bit".

Syntax
BOOL SetTagByte(Tag Tag_Name, BYTE value);

Parameters
- **Tag_Name**
  name of the tag
- **value**
  Value of the tag in the data type "BYTE"

Return value
- **TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.
- **FALSE**
An error has occurred.

See also
- SetTagWord example (Page 1577)
- Principle of the SetTag functions (Page 1467)

SetTagChar

Function
Sets the value of a tag of the data type "8-bit text tag" or "16-bit text tag".

Parameter
- **Tag_Name**
  Name of the tag
Value
Value of the tag in the data type "LPSTR"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagChar example (Page 1570)
Principle of the SetTag functions (Page 1467)

SetTagDateTime

Function
Sets the value of a tag of data type "Date/Time".

Syntax
BOOL SetTagDateTime(Tag Tag_Name, SYSTEMTIME value);

Parameter

Tag_Name
Name of the tag

value
Value of the tag in the data type "Date/Time".

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.
SetTagDouble

Function
Defines the value of a tag of the data type "64-bit floating point value".

Syntax
BOOL SetTagDouble(Tag Tag_Name, double value);

Parameters
Tag_Name
name of the tag

value
Value of the tag in the data type "double"

Return value
TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagFloat example (Page 1571)
Principle of the SetTag functions (Page 1467)

SetTagDWord

Function
Sets the value of a tag of the data type "unsigned 32 Bit".

Syntax
BOOL SetTagDWord(Tag Tag_Name, DWORD value);
Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "DWORD"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagWord example (Page 1577)
Principle of the SetTag functions (Page 1467)

SetTagFloat

Function

Defines the value of a tag of the data type "32-bit floating point value".

Syntax

BOOL SetTagFloat(Tag Tag_Name, float value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "float"
**Return value**

**TRUE**  
The function itself has been completed without any errors.  
However, no test is made as to whether the tag could be written without errors.

**FALSE**  
An error has occurred.

**See also**
- SetTagFloat example (Page 1571)  
- Principle of the SetTag functions (Page 1467)

**SetTagRaw**

**Function**

Sets the value of a tag of the data type "Raw data type".

**Syntax**

```c
BOOL SetTagRaw(Tag Tag_Name, BYTE* pValue, DWORD size);
```

**Parameters**

- **Tag_Name**  
  name of the tag

- **pValue**  
  The pointer to a byte field which contains the value of the raw data tag

- **size**  
  Size of the byte field in bytes

**Return value**

**TRUE**  
The function itself has been completed without any errors.  
However, no test is made as to whether the tag could be written without errors.

**FALSE**  
An error has occurred.
See also

SetTagRaw example (Page 1574)
Principle of the SetTag functions (Page 1467)

SetTagSByte

Function

Sets the value of a tag of the data type “signed 8 bit”.

Syntax

BOOL SetTagSByte(Tag Tag_Name, signed char value);

Parameters

Tag_Name
name of the tag

value
The value of the tag in the data type “signed char”

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagSByte example (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagSDWord

Function

Sets the value of a tag of the data type “signed 32 bit”.
Syntax

BOOL SetTagSDWord(Tag Tag_Name, long value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "long"

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also

SetTagSByte example (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagSWord

Function

Sets the value of a tag of the data type "signed 16 bit".

Syntax

BOOL SetTagSWord(Tag Tag_Name, short value);

Parameters

Tag_Name
name of the tag

value
Value of the tag in the data type "short"
size
Size of the byte field in bytes

Return value

TRUE
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

FALSE
An error has occurred.

See also
SetTagSByte example (Page 1576)
Principle of the SetTag functions (Page 1467)

SetTagValue

Function
Enables the transfer of a value in the form of a variant and sets the pointer to the value of the
data type "Variant".

Syntax

BOOL SetTagValue(LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError);

Parameters

lpdmVarKey
Pointer to a structure of the data type "DM_VARKEY"

lpdmValue
Pointer to the value of data type "Variant". A description of the data type VARIANT can be
found in the associated documentation.

dwState
Tag status which is returned after the function has been run.

lpdmError
Pointer to the structure which contains the error description
Return value

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

**FALSE**
An error has occurred.

See also

- Principle of the SetTag functions (Page 1467)
- Structure definition CMN_ERROR (Page 1601)
- Tag statuses (Page 1598)
- Structure definition DM_VAR_UPDATE_STRUCT (Page 1603)
- Structure definition DM_VARKEY (Page 1605)

SetTagWord

Function

Sets the value of a tag of the data type "unsigned 16 Bit".

Syntax

```c
BOOL SetTagWord(Tag Tag_Name, WORD value);
```

Parameters

- **Tag_Name**
  name of the tag
- **value**
  Value of the tag in the data type "WORD"

Return value

**TRUE**
The function itself has been completed without any errors.
However, no test is made as to whether the tag could be written without errors.

**FALSE**
An error has occurred.
DeactivateRTProject

Function
Deactivates the activated project.

Note
If Runtime is exited on a server or client this applies only to the respective computer.
An activated project for which the WinCC Explorer has not been started must be closed with the internal function "ExitWinCC".
If the activated project was exited with the internal function "DeactivateRTProject" the WinCC project remains open in the background. To close this project, the WinCC Explorer must be opened and then be closed by means of the menu commands "File" > "Exit".

Syntax
BOOL DeactivateRTProject();

Return value
TRUE
The function has been completed without any errors.
FALSE
An error has occurred.

See also
DeactivateRTProject example (Page 1517)

ExitWinCC

Function
Deactivates Runtime and exits WinCC on the computer executing the function.

Note
If Runtime is exited on a server or client this applies only to the respective computer.

An activated project for which the WinCC Explorer has not been started must be closed with the internal function "ExitWinCC".

If the activated project was exited with the internal function "DeactivateRTProject" the WinCC project remains open in the background. To close this project, the WinCC Explorer must be opened and then be closed by means of the menu commands "File" > "Exit".

Syntax
BOOL ExitWinCC ();

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also
ExitWinCC example (Page 1517)
GetLanguage

Function

Determines the current Runtime language.

Syntax

DWORD GetLanguage();

Return value

The current Runtime language with the associated language identifier is returned.

Note

You can find a comprehensive "Language code" table in the "Basic Principles of VBScript" documentation under the index entry "Language code".

See also

GetLanguage example (Page 1525)

InquireLanguage

Function

Determines all languages configured in the text library for the runtime.
Use dwCount to specify where the number of determined language IDs is to be stored.

Syntax

DWORD* InquireLanguage(DWORD* dwCount);

Parameters

dwCount

Pointer to the number of determined language IDs
Return value

The configured languages with the associated language identifiers are returned.

Note

You can find a comprehensive "Language code" table in the "Basic Principles of VBScript" documentation under the index entry "Language code".

See also

InquireLanguage example (Page 1557)

SetLanguage

Function

Changes the language setting in Runtime.

Syntax

BOOL SetLanguage(DWORD dwLocaleID);

Parameters

dwLocaleID
Language ID of the language to be set

Return value

TRUE
The function has been completed without any errors.

FALSE
An error has occurred.

See also

Language ID (Page 1596)
SetLanguage example (Page 1565)
FillDiagnoseInTags

Function
Activates or deactivates the storage of diagnostic information in tags.
As filling the tags is an additional load for the system, it should only be activated temporarily
for diagnostic information.

Syntax
void FillDiagnoseInTags(BOOL bfill);

Parameters
bFill
Storage of diagnostic information in tags on/off

TRUE  Activate supply of diagnostic tags
FALSE  Deactivate supply of diagnostic tags

Diagnostic tags of GlobalScript

@SCRIPT_COUNT_TAGS
This tag contains the current number of tags requested via Script.

@SCRIPT_COUNT_REQUEST_IN_QUEUES
This tag contains the current number of jobs.

@SCRIPT_COUNT_ACTIONS_IN_QUEUES
This tag contains the current number of actions.

GetServerTagPrefix

Function
To be able to access tags of the respective server from a WinCC client in a distributed system,
the tag names must be supplemented with the server prefix.
If the tags are accessed by means of the functions GetTagxx or SetTagxx, the required addition
is made by the script control.
If WinCC API functions are used for accessing, the tag names have to be supplemented by
the user. The GetServer TagPrefix function provides the required prefixes.
One pointer each of the “char” type to ServerPrefix, TagPrefix and WindowPrefix is returned.
The user must neither change the memory (also no strcat) nor release it.
Syntax

```c
void GetServerTagPrefix(char** ppszServerPrefix, char** ppszTagPrefix, char** ppszWindowPrefix);
```

Parameters

- `ppszServerPrefix`
  Pointer to a pointer referring to the server prefix
- `ppszTagPrefix`
  Pointer to a pointer referring to the tag prefix
- `ppszWindowPrefix`
  Pointer to a pointer referring to the window prefix

See also

- GetServerTagPrefix example (Page 1538)

TraceText

Function

The value defined in `<Parameter>` is recorded in APDiag if the specified diagnostic level has been reached.

Syntax

```c
void TraceText(DWORD dwTraceLevel, char* pszFormat, <Parameter>);
```

Parameters

- `dwTraceLevel`
  Diagnostic level
- `pszFormat`
  Output format (according to printf function)
<Parameter>
Value to be reported

---

Note
The parameterization dialog for this function provides the selection of tags, graphic objects and pictures.

---

TraceTime

Function
The value defined in <Parameter> is recorded in APDiag if the specified diagnostic level has been reached.
In addition, the time since the AP start of diagnosis is output in milliseconds to enable performance measurements.

Syntax

```c
void TraceTime(DWORD dwTraceLevel, char* pszFormat, <Parameter>);
```

Parameters

- **dwTraceLevel**
  Diagnostic level

- **pszFormat**
  Output format (according to printf function)

- **<Parameter>**
  Value to be reported

---

Note
The parameterization dialog for this function provides the selection of tags, graphic objects and pictures.
2.15.4   Examples

2.15.4.1   Examples - A to G

AcknowledgeMessage example

{
    //Acknowledge the AlarmLogging message which is selected
    AcknowledgeMessage(GetTagWord("U08i_MsgNr"));
}

Specify the message number to be acknowledged. It is read from a tag.

DeactivateRTProject example

{
    //deactivate the runtime
    DeactivateRTProject ();
}

This function deactivates WinCC Runtime.

ExitWinCC example

{
    //exit wincc
    ExitWinCC ();
}

This function exits WinCC.
2.15.4.2 Examples - GetAlarmHigh to GetPropChar

GetAlarmHigh example

```c
{
    double dAlarmHigh;
    //Get the Alarm High Limit
    dAlarmHigh = GetAlarmHigh(lpszPictureName,"Bar1");

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the function GetAlarmHigh:

"lpszPictureName" is the name of the picture in which the object was configured.

"Bar1" is the name of the object.

1. Read out the upper alarm limit and temporarily store it in dAlarmHigh.
2. Executing user-defined code for processing return values.

GetBackColor example

```c
{
    long int bk_color;

    //Get the background color
    bk_color = GetBackColor(lpszPictureName,"StatischerText1");

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the function GetBackColor:

"lpszPictureName" is the name of the picture in which the object was configured.

"StaticText1" is the name of the object.

1. Read out the current background color and temporarily store it in bk_color.
2. Executing user-defined code for processing return values.
GetBorderStyle example

{
    long int lstyle;

    //Get the current border style
    lstyle = GetBorderStyle(lpszPictureName,"Rectangle1");

    //User defined code where the
    //user can do something with the return value
    ... }

Parameters of the function GetBorderStyle:
"lpszPictureName" is the name of the picture in which the object was configured.
"Rectangle1" is the name of the object.

1. Read out the current line style of the object and temporarily store it in lstyle.
2. Executing user-defined code for processing return values.

GetFilling example

{
    BOOL bfilling;

    //Get the actual state of dynamic filling
    bfilling = GetFilling(lpszPictureName,"Rectangle1");

    if(bfilling)
    {
        // User defined code if the
        // dynamic filling is activated
        ... }
    Else
    {
        // User defined code if the
        // dynamic filling is deactivated
        ... }
}

Parameters of the function GetFilling:
"lpszPictureName" is the name of the picture in which the object was configured.
"Rectangle1" is the name of the object.

1. Read out whether dynamic filling is activated or not and temporarily store in bfilling.
2. Executing user-defined code, depending on the return value of the function.

**GetFillingIndex example**

```c
{
    long int filling_index;
    //Get the actual filling index of the object
    filling_index = GetFillingIndex(lpszPictureName,"Rectangle1");
    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the function GetFillingIndex:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Rectangle1" is the name of the object.

1. Read out the current fill level of the object and temporarily store it in filling_index.
2. Executing user-defined code for processing return values.

**GetFillStyle example**

```c
{
    long int lstyle;
    //Get the current fill style
    lstyle = GetFillStyle(lpszPictureName,"Rectangle1");
    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the function GetFillStyle:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Rectangle1" is the name of the object.
1. Read out the current fill pattern of the object and temporarily store it in lstyle.
2. Executing user-defined code for processing return values.

GetFlashBackColor example

```c
{
    BOOL bflash_col;

    // Get if the flashing is on or off
    bflash_col = GetFlashBackColor(lpszPictureName,"Group1");

    if(bflash_col)
    {
        // User defined code if the
        // flashing is activated
        ...
    }
    Else
    {
        // User defined code if the
        // flashing is deactivated
        ...
    }
}
```

Parameters of the function GetFlashBackColor:
"lpszPictureName" is the name of the picture in which the object was configured.
"Group1" is the name of the object.

1. Read out whether flashing of the background color is activated or not and temporary store in bflash_col.
2. Executing user-defined code, depending on the return value of the function.

GetFlashBackColorOn example

```c
{
    long int flashcol_on;

    // Get the BackFlashColor
    flashcol_on = GetBackFlashColorOn(lpszPictureName,"Group1");

    // User defined code where the
    // user can do something with the return value
    ...
}
```
Parameters of the GetBackFlashColorOn function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Group1" is the name of the object.

1. Read out the background flash color for the "On" status of the object and temporarily store it in flashcol_on.
2. Executing user-defined code for processing return values.

GetFlashRateFlashPic example

```

lFlashRate = GetFlashRateFlashPic(lpszPictureName,"StatusDisplay1");
```

Parameters of the GetFlashRateFlashPic function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Status display1" is the name of the object.

1. Read out the flash frequency of the object and temporarily store it in lFlashRate.
2. Executing user-defined code for processing return values.
GetFocus example

```c
char* pszValue = NULL;
char szValue[_MAX_PATH+1];

//Get the Object which has the focus
pszValue = Get_Focus();

//Copy the string
if(pszValue != NULL)
{
  strncpy(szValue,pszValue,_MAX_PATH);
}
//User defined code where the
//user can do something with the return value
...
```

1. Read out on which object the focus is and temporarily store in pszValue.
2. If a valid value has been returned, store the return value of the function in the local string szValue. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.

GetFontBold example

```c
BOOL bbold;

//Get if the text is bold
bbold = GetFontBold(lpszPictureName,"StaticText1");

if(bbold)
{
  // User defined code if the
  // font is bold
  ...
}
else
{
  // User defined code if the
  // font is not bold
  ...
}
```

Parameters of the GetBackColor function
"lpszPictureName" is the name of the picture in which the object was configured.
"StaticText1" is the name of the object.

1. Read out whether the text is in bold or not and temporarily store in bbold.
2. Executing user-defined code, depending on the return value of the function.

GetFontSize example

{ long int fontsize;

//Get the actual Font size
fontsize = GetFontSize(lpszPictureName,"StaticText1");

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetFontSize function:
"lpszPictureName" is the name of the picture in which the object was configured.
"StaticText1" is the name of the object.

1. Read out the current font size and temporarily store it in fontsize.
2. Executing user-defined code for processing return values.

GetHeight example

{ long lHeight;

//Get the height of the object
lHeight = GetHeight(lpszPictureName,"WinCCLogo");

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetHeight function:
"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.
1. Read out the current height of the object and temporarily store it in lHeight.
2. Executing user-defined code for processing return values.

GetHiddenInput example

```c
{
 BOOL bHiddenInput;

 //Get the state of hidden input
 bHiddenInput = GetHiddenInput(lpszPictureName,"IOField1");

 if(bHiddenInput)
 {
   // User defined code if the
   // hidden input is activated
   ...
 }
 Else
 {
   // User defined code if the
   // hidden input is activated
   ...
 }
}
```

Parameters of the GetHiddenInput function:

"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.

1. Read out whether the text is in bold or not and temporarily store in bHiddenInput.
2. Executing user-defined code, depending on the return value of the function.

GetLanguage example

```c
{
 DWORD rt_language;

 //Get the current language
 rt_language = GetLanguage();

 //User defined code where the
 //user can do something with the return value
 ...
}
```
1. Read out the current Runtime language and temporarily store it in rt_language.
2. Executing user-defined code for processing return values.

GetLeft example

{
long lPos;

//Get the x-position of the object
lPos = GetLeft(lpszPictureName,"WinCCLogo");

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetLeft function:
"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.

1. Read out the current X position of the object and temporarily store it in lPos.
2. Executing user-defined code for processing return values.

GetLink example

{
LINKINFO linkinfo;

//Get the linked Tag
GetLink(lpszPictureName,"Bar1","Process",&linkinfo);

// linkinfo.szLinkName is the tag name
// linkinfo.dwCycle is the update cycle
// linkinfo.LinkType is the type of the connection

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetLink function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.
"Process" is the property connected to a tag.
"&linkinfo" is the address of the linkinfo structure.

1. Fills the passed linkinfo structure with the tag connection information.
2. Executing user-defined code, depending on the return value of the function.

GetLinkedVariable example

```c
{
    char* pszVarName = NULL;
    char szVarName[_MAX_PATH+1];

    //Get the TagName
    pszVarName = GetLinkedVariable("gs_stand_graph_00","StaticText6","Visible");

    //Copy the string
    if (strcmp (pszVarName,"")!= 0)
    {
        strncpy(szVarName,pszVarName,_MAX_PATH);
    }
    else printf("Attribute 'visible' is not made dynamic\n");

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the GetLinkedVariable function:
"gs_stand_graph_00" is the name of the picture in which the object was configured.
"StaticText6" is the name of the object.
"Visible" is the property connected to a tag.

1. Temporarily store the return value of the GetLinkedVariable function in pszVarName.
2. If a valid value has been returned, store the return value in szVarName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.
GetLocalPicture example

```
char* pszPicName = NULL;
char szPicName[_MAX_PATH+1];

// Get the Local Picture
pszPicName = GetLocalPicture(lpszPictureName);

// Copy the string
if (pszPicName != NULL)
{
    strncpy(szPicName,pszPicName,_MAX_PATH);
}

// User-defined code where the user can do something with the return value
...
```

1. Temporarily store the return value of the GetLocalPicture function in pszPicName.
2. If a valid value has been returned, store the return value in szPicName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.

GetMarker example

```
BOOL bmarker;

// Get the state of the Marker
bmarker = GetMarker(lpszPictureName,"Bar1");

if(bmarker)
{
    // User defined code if the marker is activated
    ...}
Else
{
    // User defined code if the marker is deactivated
    ...}
```

Parameters of the GetMarker function:

"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.

1. Read out whether the marker is displayed or not and temporarily store in bmarker.
2. Executing user-defined code, depending on the return value of the function.

GetOutputValueDouble example

```c
{
    double doutput;

    //Get the output value of IO Field 1
    doutput = GetOutputValueDouble(lpszPictureName,"IOField1");

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the GetOutputValueDouble function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "IOField1" is the name of the object.

1. Read out the output value and temporarily store it in doutput.
2. Executing user-defined code for processing return values.

GetParentPicture example

```c
{
    char* pszPicName = NULL;
    char szPicName[_MAX_PATH+1];

    //Get the parent picture
    pszPicName = GetParentPicture(lpszPictureName);

    //Copy the string
    if (pszPicName != NULL)
    {
        strncpy(szPicName,pszPicName,_MAX_PATH);
    }

    //User defined code where the
    //user can do something with the return value
    ...
}
```
1. Temporarily store the return value of the GetParentPicture function in pszPicName.
2. If a valid value has been returned, store the return value in szPicName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.

GetPictureDown example

```
{  
    char* pszPicName = NULL;
    char szPicName[_MAX_PATH+1];

    //Get the current picture name
    pszPicName = GetPictureDown(lpszPictureName,"Roundbutton1");
    if (pszPicName != NULL) 
    {  
        //Copy the string
        strncpy(szPicName,pszPicName,_MAX_PATH);
    }
    //User defined code where the
    //user can do something with the return value
    ...  
}
```

Parameters of the GetPictureDown function:

"lpszPictureName" is the name of the picture in which the object was configured.

"RoundButton1" is the name of the object.

1. Read out the picture name of the picture displayed in round button 1 and temporarily store it in pszPicName.
2. If a valid value has been returned, store the return value of the function in the local string szPicName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.
GetPictureName example

{
char* pszPictureName = NULL;
char szPictureName[_MAX_PATH + 1];

//Get the current PictureName
pszPictureName = GetPictureName(lpszPictureName,"GraphicObject1");

if(pszPictureName != NULL)
{
//copy the string
strncpy(szPictureName,pszPictureName, _MAX_PATH);
}
//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetPictureName function:
"lpszPictureName" is the name of the picture in which the object was configured.
"GraphicObject1" is the name of the object.

1. Read out the picture name of the picture displayed in graphic object 1 and temporarily store it in pszPictureName.
2. If a valid value has been returned, store the return value of the function in the local string szPictureName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.
**GetPictureUp example**

```c
char* pszPicName = NULL;
char szPicName[_MAX_PATH+1];

//Get the current picture name
pszPicName = GetPictureUp(lpszPictureName,"Roundbutton1");

if (pszPicName != NULL)
{
    //Copy the string
    strncpy(szPicName,pszPicName,_MAX_PATH);
}

//User defined code where the
//user can do something with the return value
...}
```

Parameters of the GetPictureUp function:

"lpszPictureName" is the name of the picture in which the object was configured.

"RoundButton1" is the name of the object.

1. Read out the picture name of the picture displayed in round button 1 and temporarily store it in pszPicName.
2. If a valid value has been returned, store the return value of the function in the local string szPicName. A maximum of _MAX_PATH characters is stored.
3. Executing user-defined code for processing return values.

**GetPosition example**

```c
long int lpos;

//Get the actual position of the Slider
lpos = GetPosition(lpszPictureName,"Control1");

//User defined code where the
//user can do something with the return value
...}
```

Parameters of the GetPosition function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Control1" is the name of the object.

1. Read out the current slider position and temporarily store it in lpos.
2. Executing user-defined code for processing return values.

GetPropBOOL example

```c
BOOL bProp;

//Get the property Visible
bProp = GetPropBOOL("gs_graph_eafield","IOField1","Visible");

if(bProp)
{
   // User defined code if the
   // object is visible
   ...
}
else
{
   // User defined code if the
   // object is not visible
   ...
}
```

Parameters of the GetVisible function:

"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"Visible" is the object property.

1. Read out whether the object is visible or not and temporarily store in bProp.
2. Executing user-defined code, depending on the return value of the function.
GetPropChar example

{
char* pszProp = NULL;
char szProp[14];

//Get the property Tooltiptext
pszProp = GetPropChar("lpszPictureName","EAFeld1","Tooltiptext");

if(pszProp != NULL)
{
   //Copy the string
   strncpy(szProp,pszProp,13);
   //User defined code where the
   //user can do something with the return value
   ...
}

Parameters of the GetPropChar function:
"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"Tooltiptext" is the object property.

1. Read out the tooltip text of the object and temporarily store it in pszProp.
2. If a valid value has been returned, store the return value of the function in the local string szProp. A maximum of 13 characters is stored.
3. Executing user-defined code for processing return values.

2.15.4.3 Examples - GetRangeMax to GetWidth

GetRangeMax example

{
long int lrange;

//Get the upper scale Limit
lrange = GetRangeMax(lpszPictureName,"Controll");

//User defined code where the
//user can do something with the return value
...
}
Parameters of the `GetRangeMax` function:

"lpzPictureName" is the name of the picture in which the object was configured.
"Control1" is the name of the object.

1. Read out the current upper limit of the object and temporarily store it in `lrange`.
2. Executing user-defined code for processing return values.

**GetRangeMin example**

```c
{
    long int lrange;

    //Get the lower scale Limit
    lrange = GetRangeMin(lpszPictureName,"Control1");

    //User defined code where the
    //user can do something with the return value
    ...  
}
```

Parameters of the `GetRangeMin` function:

"lpzPictureName" is the name of the picture in which the object was configured.
"Control1" is the name of the object.

1. Read out the current lower limit of the object and temporarily store it in `lrange`.
2. Executing user-defined code for processing return values.
Beispiel GetScaling

```c
{  BOOL bscaling;

  // Get the Scaling state  
  bscaling = GetScaling(lpszPictureName,"Bar1");

  if (bscaling)
  {
    // User defined code if the  
    // bar object has an additional scale  
    ...
  }
  Else
  {
    // User defined code if the  
    // bar object has no additional scale  
    ...
  }
}

Parameters of the GetScaling function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.

1. Read out whether the scale of the bar is displayed or not and temporarily store in bscaling.
2. Executing user-defined code, depending on the return value of the function.
GetServerTagPrefix example

```c
{
    char* pszServerPrefix;
    char* pszTagPrefix;
    char* pszWindowPrefix;
    int nServerPrefixLen = 0;
    int nTagPrefixLen = 0;
    int nTagLen = 0;
    char myTagName[MAX_DM_VAR_NAME+1];

    //Initialize the return value
    memset(myTagName,0,MAX_DM_VAR_NAME + 1);

    //Get the serverprefix the tagprefix and the windowprefix
    GetServerTagPrefix(&pszServerPrefix, &pszTagPrefix, &pszWindowPrefix);

    //If a serverprefix exists
    if (pszServerPrefix)
    {
        //Get the length of the string
        nServerPrefixLen = strlen(pszServerPrefix);
    }
    Else
    {
        printf("No server prefix was returned.");
        return;
    }  

    //If a tagprefix exists
    if (pszTagPrefix)
    {
        //Get the length of the string
        nTagPrefixLen = strlen(pszTagPrefix);
    }  

    //Get the length of the tag
    nTagLen = strlen("TagName");

    //Check if the length of the
    //ServerPrefix+TagPrefix+VarName + the double points < MAX_DM_VAR_NAME)
    if (nServerPrefixLen + nTagPrefixLen + nTagLen+2 < MAX_DM_VAR_NAME)
    {
        sprintf(myTagName,"%s:%s\%s",pszServerPrefix,pszTagPrefix,"TagName");
        //User defined code where the
        //user can do something with the return value
        ...
    }
    Else
    {
        printf("The resulting string is too long.");
        return;
    }
}
```
1. Initialize the myTagName tag.
2. Read out the server prefix, the tag prefix and the window prefix.
3. If no server prefix has been returned, a text is output and the function is terminated.
4. If a server prefix has been returned, determine its length and temporarily store it in nServerPrefixLen.
5. If a tag prefix has been returned, determine its length and temporarily store it in TagPrefixLen.
6. Determine the length of the tag name and temporarily store it in nVarLen.
7. If the length permitted for tag names is exceeded a text is output and the function is terminated.
8. If the length permitted for tag names is not exceeded, the tag name required for a client environment is compiled.
9. Executing user-defined code for processing return values.

GetServerTagPrefix example

```
{  
  char* pszServerPrefix;
  char* pszTagPrefix;
  char* pszWindowPrefix;

  //Get the serverprefix and the tagprefix  
  GetServerTagPrefix(&pszServerPrefix, &pszTagPrefix, &pszWindowPrefix);
  //User defined code where the  
  //user can do something with the return value
  ...
}
```

Parameters of the GetServerTagPrefix function:
"pszServerPrefix" is the tag which is written into the server prefix.
"pszTagPrefix" is the tag which is written into the tag prefix.
"pszWindowrPrefix" is the tag which is written into the window prefix.

1. Read out the server prefix, the tag prefix and the window prefix.
2. The pszServerPrefix tag contains the returned server prefix.
3. The pszTagPrefix tag contains the returned tag prefix.
4. The pszWindowPrefix tag contains the returned window prefix.
5. Executing user-defined code for processing return values.
GetTagBit example

```c
{  
    BOOL bstate;

    // Get the current state of the tag
    bstate = GetTagBit("gs_tag_bit");

    if(bstate)
    {
        // User defined code if the
        // value of the tag is true
        ...
    }
    else
    {
        // User defined code if the
        // value of the tag is false
        ...
    }
}
```

Parameters of the GetTagBit function

"gs_tag_bit" is the name of the tag.

1. Read out the value of the tag and temporarily store it in bstate.
2. Executing user-defined code, depending on the return value of the function.
GetTagBitStateQC example

{
DWORD dwState;
DWORD dwQC;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitState QCWait("gs_tag_bit",&dwState,&dwQC);

//Create a string which includes the tag value
if (bValue)
{
   // User defined code if the
   // value of the tag is true
   ...
}
else
{
   // User defined code if the
   // value of the tag is false
   ...
}
}

Parameters of the GetTagBitStateQC function:
"gs_tag_bit" is the name of the tag.
"&dwState" is the address of the tags in which the tag status is to be stored.
"&dwQC" is the address of the tag in which the quality code is to be stored.

1. Read out the value of the tag and temporarily store it in bValue. The function puts the tag status in dwState and the quality code in dwQC.
2. Executing user-defined code, depending on the return value of the function.
GetTagBitStateWait example

{
    DWORD dwstate;
    BOOL bValue;

    dwstate = 0xFFFFFFFF;

    //Get the tag value
    //dwstate is the tag state
    bValue = GetTagBitStateWait("gs_tag_bit", &dwstate);

    //Create a string which includes the tag value
    if (bValue)
    {
        // User defined code if the
        // value of the tag is true
        ...
    }
    else
    {
        // User defined code if the
        // value of the tag is false
        ...
    }
}

Parameters of the GetTagBitStateWait function:
"gs_tag_bit" is the name of the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Read out the value of the tag and temporarily store it in bstate. The function puts the tag status in dwstate.
2. Executing user-defined code, depending on the return value of the function.
GetTagChar example

{
    char* pszValue = NULL;
    char szValue[13];

    //Get the current value of the tag
    pszValue = GetTagChar("gs_tag_char");

    if(pszValue != NULL)
    {
        //Copy the string
        strncpy(szValue,pszValue,12);
        //User defined code where the
        //user can do something with the return value
        ...
    }
}

Parameters of the GetTagChar function:
"gs_tag_char" is the name of the tag.

1. Reading the value of the tag and temporarily storing in pszValue.
2. If a valid value has been returned, store the return value of the function in the local string szValue. A maximum of 12 characters is stored.
3. Executing user-defined code for processing return values.
GetTagCharStateQCWait example

```c
{
  DWORD dwState;
  DWORD dwQC;
  char* pszRetValue = NULL;
  char szRetValue[13];
  
  dwState = 0xFFFFFFFF;
  
  //Get the tag value
  pszRetValue = GetTagCharStateQCWait("gs_tag_char", &dwState, &dwQC);
  
  if (pszRetValue != NULL) {
    //Copy the string
    strncpy(szRetValue, pszRetValue, 12);
  }
  
  //User defined code where the
  //user can do something with the return value
  ...
}
```

Parameters of the GetTagCharStateQCWait function:

- "gs_tag_char" is the name of the tag.
- "&dwState" is the address of the tags in which the tag status is to be stored.
- "&dwQC" is the address of the tag in which the quality code is to be stored.

1. Read out the value of the tag and temporarily store it in pszRetValue. The function puts the tag status in dwState and the quality code in dwQC.
2. If a valid value has been returned, store the return value of the function in the local string szRetValue. A maximum of 12 characters is stored.
3. Executing user-defined code for processing return values.
Beispiel GetTagCharStateWait

```c
{ 
DWORD dwstate;
char szValue[11];
char* pszRetValue = NULL;
char szRetValue[13];

dwstate = 0xFFFFFFFF;
//Get the tag value
//dwstate is the tag state
pszRetValue = GetTagCharStateWait("gs_tag_char", &dwstate);

if (pszRetValue != NULL)
{
    //Copy the string
    strncpy(szRetValue, pszRetValue, 12);
}
//User defined code where the
//user can do something with the return value
... 
}
```

Parameters of the GetTagCharStateWait function:
"gs_tag_char" is the name of the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Read out the value of the tag and temporarily store it in pszRetValue. The function puts the tag status in dwstate.
2. If a valid value has been returned, store the return value of the function in the local string szRetValue. A maximum of 12 characters is stored.
3. Executing user-defined code for processing return values.

GetTagFloat example

```c
{ 
float fValue;

//Get the current value of the tag
fValue = GetTagFloat("gs_tag_float");

//User defined code where the
//user can do something with the return value
... 
}
```
Parameters of the GetTagFloat function:
"gs_tag_float" is the name of the tag.

1. Read out the value of the tag and temporarily store it in fValue.
2. Executing user-defined code for processing return values.

GetTagFloatStateQCWait example

```c

{  DWORD dwState;
  DWORD dwQC;
  float fValue;

  dwState = 0xFFFFFFFF;

  //Get the tag value
  fValue = GetTagFloatStateQCWait("gs_tag_float", &dwState, &dwQC);

  //User defined code where the
  //user can do something with the return value
  ...
}
```

Parameters of the GetTagFloatStateQCWait function:
"gs_tag_float" is the name of the tag.
"&dwState" is the address of the tags in which the tag status is to be stored.
"&dwQC" is the address of the tag in which the quality code is to be stored.

1. Read out the value of the tag and temporarily store it in fValue. The function puts the tag status in dwState and the quality code in dwQC.
2. Executing user-defined code for processing return values.
GetTagFloatStateWait example

{
    DWORD dwstate;
    float fValue;

    dwstate = 0xFFFFFFFF;
    //Get the tag value
    //dwstate is the tag state
    fValue = GetTagFloatStateWait("gs_tag_float", &dwstate);

    //User defined code where the
    //user can do something with the return value
    ...  
}

Parameters of the GetTagFloatStateWait function:
"gs_tag_float" is the name of the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Read out the value of the tag and temporarily store it in fValue. The function puts the tag status in dwstate.
2. Executing user-defined code for processing return values.
GetTagMultiStateQCWait example

{
#define DATA_SIZE 5
DWORD dwState[DATA_SIZE];
DWORD dwQC[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2;
char* szValue3;
double dblValue4;
WORD lValue5;

//Set the tags
GetTagMultiStateQCWait(dwState,dwQC,"%d%d%s%f%d",
   "gs_tag_bit",&lValue1,
   "gs_tag_SByte",&lValue2,
   "gs_tag_char",&szValue3,
   "gs_tag_float",&dblValue4,
   "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetTagMultiStateWait function:
"dwState" is the DWord-Array, in which the tag statuses are stored.
"dwQC" is the DWord-Array, in which the quality codes are stored.
"%d%d%s%f%d" are the type descriptions of the tags to be read.
"gs_tag_bit" is the tag to be read.
"&lValue1" is the address of the tags in which the value of the tags gs_tag_bit should be stored.
"gs_tag_SByte" is the tag to be read.
"&lValue2" is the address of the tags in which the value of the tags gs_tag_SByte should be stored.
The other parameters are to be handled in the same way as those described previously.

1. Creating a DWord-Array with the required size (Number of tags).
2. Reading and storing the values of the tags. The value of the tags gs_tag_bit is stored temporarily in lValue1. The value of the tags gs_tag_SByte is stored temporarily in lValue2, etc.
3. Executing user-defined code for processing return values.
GetTagMultiStateWait example

```c
{
#define DATA_SIZE 5
DWORD dwData[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2;
char* szValue3;
double dblValue4;
WORD lValue5;

//Set the tags
GetTagMultiStateWait(dwData,"%d%d%s%f%d",
    "gs_tag_bit",&lValue1,
    "gs_tag_SByte",&lValue2,
    "gs_tag_char",&szValue3,
    "gs_tag_float",&dblValue4,
    "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the return value
...
}
```

Parameters of the GetTagMultiStateWait function:

"dwData" is the DWord-Array, in which the tag statuses are stored.
"%d%d%s%f%d" are the type descriptions of the tags to be read.
"gs_tag_bit" is the tag to be read.
"&lValue1" is the address of the tags in which the value of the tags gs_tag_bit should be stored.
"gs_tag_SByte" is the tag to be read.
"&lValue2" is the address of the tags in which the value of the tags gs_tag_SByte should be stored.

The other parameters are to be handled in the same way as those described previously.

1. Creating a DWord-Array with the required size (Number of tags).
2. Reading and storing the values of the tags. The value of the tags gs_tag_bit is stored temporarily in lValue1. The value of the tags gs_tag_SByte is stored temporarily in lValue2, etc.
3. Executing user-defined code for processing return values.
GetTagMultiWait example

DWORD dwVar1Value;
char* szVar2Value;
//Memory for the tag value is
//created by teh function with SysMalloc
double dbVar3Value;

BOOL ok;

ok=GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
   "Ernie_char", &szVar2Value,
   "Ernie_double", &dbVar3Value);

printf("Word %d, String %s, Double %f\r\n",
   dwVar1Value, szVar2Value, dbVar3Value);

GetTagPrefix example

{
char* pszTagPrefix = NULL;
char szTagPrefix[7];

//Get the current tag prefix
pszTagPrefix = GetTagPrefix(lpszPictureName,"PicWindow1");

if(pszTagPrefix != NULL)
{
   //Copy the string
   strncpy(szTagPrefix,pszTagPrefix,6);
}
   //User defined code where the
   //user can do something with the return value
...
}

Parameters of the GetTagPrefix function:
"lpszPictureName" is the name of the picture in which the object was configured.
"PictureWindow1" is the name of the object.

1. Read out the current tag prefix of picture window 1 and temporarily store it in pszTagPrefix.
2. If a valid value has been returned, store the return value of the function in the local string
   szTagPrefix. A maximum of 6 characters is stored.
3. Executing user-defined code for processing return values.
GetTagRaw example

{
    #define DATA_SIZE 3
    BYTE byData[DATA_SIZE];

    //Get the current values of the tag
    GetTagRaw("gs_tag_raw", byData, DATA_SIZE);

    //Use the values received in the array byData
    ...
}

Parameters of the GetTagRaw function:
"gs_tag_raw" is the name of the tag.
"byData" is the byte array in which the values of the raw data tags will be stored.
"DATA_SIZE" is the number of values that will be read.

1. Reading the values of the tags and temporarily storing in byData.
2. Executing user-defined code for processing return values.

GetTagRawStateQCWait example

{
    #define DATA_SIZE 3
    DWORD dwState;
    DWORD dwQC;
    BYTE byData[DATA_SIZE];

    dwState = 0xFFFFFFFF;

    //Get the values of the tag
    GetTagRawStateQCWait("gs_tag_raw", byData, DATA_SIZE, &dwState, &dwQC);

    //User defined code where the
    //user can do something with the return value
    ...
}

Parameters of the GetTagRawStateQCWait function:
"gs_tag_raw" is the name of the tag.
"byData" is the byte array in which the values of the raw data tags will be stored.
"DATA_SIZE" is the number of values that will be read.
"&dwState" is the address of the tags in which the tag status is to be stored.
"&dwQC" is the address of the tag in which the quality code is to be stored.

1. Reading the values of the tags and temporarily storing in byData.
2. Executing user-defined code for processing return values.

GetTagRawStateWait example

```c
{
#define DATA_SIZE 3
DWORD dwstate;
BYTE byData[DATA_SIZE];
char szValue[11];

//Load dwState with default values
dwstate = 0xFFFFFFFF;

//Get the values of the tag
//dwstate is the tag state
GetTagRawStateWait("gs_tag_raw",byData,DATA_SIZE,&dwstate);

//User defined code where the
//user can do something with the return value
...
}
```

Parameters of the GetTagRawStateWait function:
"gs_tag_raw" is the name of the tag.
"byData" is the byte array in which the values of the raw data tags will be stored.
"DATA_SIZE" is the number of values that will be read.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Reading the values of the tags and temporarily storing in byData.
2. Executing user-defined code for processing return values.
GetTagSByte example

```c
long lValue;

//Get the current value of the tag
lValue = GetTagSByte("gs_tag_SByte");

//User defined code where the
//user can do something with the return value
...
```

Parameters of the GetTagSByte function:
"gs_tag_SByte" is the name of the tag.

1. Read the value of the tag and temporarily store it in lValue.
2. Executing user-defined code for processing return values.

GetTagSByteStateQCWait example

```c
DWORD dwState;
DWORD dwQC;
long lValue;
dwState = 0xFFFFFFFF;

//Get the tag value
lValue = GetTagSByteStateQCWait("gs_tag_SByte", &dwState, &dwQC);

//User defined code where the
//user can do something with the return value
...
```

Parameters of the GetTagSByteState QCWait function:
"gs_tag_SByte" is the name of the tag.

"&dwState" is the address of the tags in which the tag status is to be stored.

"&dwQC" is the address of the tag in which the quality code is to be stored.

1. Read the value of the tag and temporarily store it in lValue. The function puts the tag status in dwState and the quality code in dwQC.
2. Executing user-defined code for processing return values.
GetTagSByteStateWait example

{
DWORD dwstate;
long lValue;

dwstate = 0xFFFFFFFF;
//Get the tag value
//dwstate is the tag state
lValue = GetTagSByteStateWait("gs_tag_SByte", &dwstate);

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetTagSByteStateWait function:
"gs_tag_SByte" is the name of the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Read the value of the tag and temporarily store it in lValue. The function puts the tag status in dwstate.
2. Executing user-defined code for processing return values.

GetTagWord example

{
WORD wValue;

//Get the current value of the tag
wValue = GetTagWord("gs_tag_word");

//User defined code where the
//user can do something with the return value
...
}

Parameters of the GetTagWord function:
"gs_tag_word" is the name of the tag.

1. Read out the value of the tag and temporarily store it in wValue.
2. Executing user-defined code for processing return values.
GetTagWordStateQCWait example

```c
{
    DWORD dwState;
    DWORD dwQC;
    WORD wValue;

    dwState = 0xFFFFFFFF;
    //Get the tag value
    wValue = GetTagWordStateQCWait("gs_tag_word", &dwState, &dwQC);

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the GetTagWordStateQCWait function:

- "gs_tag_word" is the name of the tag.
- "&dwState" is the address of the tags in which the tag status is to be stored.
- "&dwQC" is the address of the tag in which the quality code is to be stored.

1. Read out the value of the tag and temporarily store it in wValue. The function puts the tag status in dwState and the quality code in dwQC.
2. Executing user-defined code for processing return values.

GetTagWordStateWait example

```c
{
    DWORD dwstate;
    WORD wValue;

    dwstate = 0xFFFFFFFF;
    //Get the tag value
    //dwstate is the tag state
    wValue = GetTagWordStateWait("gs_tag_word", &dwstate);

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the GetTagWordStateWait function:

- "gs_tag_word" is the name of the tag.
- "&dwstate" is the address of the tags in which the tag status is to be stored.
1. Read out the value of the tag and temporarily store it in wValue. The function puts the tag status in dwstate.

2. Executing user-defined code for processing return values.

GetText example

```c
{
    char* pszValue = NULL;
    char szValue[13];

    //Get the Text which is actually set
    pszValue = GetText(lpszPictureName,"StaticText1");

    if(pszValue != NULL)
    {
        //Copy the string
        strncpy(szValue,pszValue,12);
    }
    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the GetText function:

"lpszPictureName" is the name of the picture in which the object was configured.

"StaticText1" is the name of the object.

1. Read out the text in the object StaticText1 and temporarily store it in pszValue.

2. If a valid value has been returned, store the return value of the function in the local string szValue. A maximum of 12 characters is stored.

3. Executing user-defined code for processing return values.

GetTop example

```c
{
    long lPos;

    //Get the y-Position of the Object
    lPos = GetTop(lpszPictureName,"WinCCLogo");

    //User defined code where the
    //user can do something with the return value
    ...
}
```
Parameters of the GetTop function:
"lpszPictureName\" is the name of the picture in which the object was configured.
"WinCCLogo\" is the name of the object.

1. Read out the current Y position of the object and temporarily store it in IPos.
2. Executing user-defined code for processing return values.

GetVisible example

```c
#include <stdbool.h>

bool bVisible;

// Get the visibility
bVisible = GetVisible(lpszPictureName, "GraphicObject1");

if (bVisible) {
    // User defined code if the
    // object is visible
    ...
} else {
    // User defined code if the
    // object is not visible
    ...
}
```

Parameters of the GetVisible function:
"lpszPictureName\" is the name of the picture in which the object was configured.
"GraphicObject1\" is the name of the object.

1. Read out whether the object is visible or not and temporarily store in bVisible.
2. Executing user-defined code, depending on the return value of the function.
GetWidth example

{
    long lWidth;

    // Get the width of the object
    lWidth = GetWidth(lpszPictureName, "WinCCLogo");

    // User defined code where the
    // user can do something with the return value
    ...
}

Parameters of the GetWidth function:

"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.

1. Read out the current width of the object and temporarily store it in lWidth.
2. Executing user-defined code for processing return values.

2.15.4.4 Examples - H to S

InquireLanguage example

{
    DWORD count;
    DWORD* language;
    int i;

    // Count the installed languages
    language = InquireLanguage(&count);

    printf("############################### INQUIRE LANGUAGE ###############################");
    // Print out the count of languages
    printf("\r\nCount Languages=%d\r\n", count);
    // Print out which languages are installed
    for (i=1; i<=count; i++)
    {
        printf("\r\n%d.language=%x", i, *language);
    }
}

1. Determine the languages configured for the runtime. In language the language IDs are temporarily stored. In count the number of languages is temporarily stored.
2. The number of determined languages is output.
3. All determined language IDs are displayed.

**ProgramExecute example**

```c
{
    //start the program calc.exe
    ProgramExecute("C:\Winnt\system32\calc.exe");
}
```

As parameter the file is to be specified with its path.

**RPTJobPreview example**

```c
{
    //Start the print preview of the specified print job
    RPTJobPreview("Documentation Text Library");
}
```

Parameters of the "RPTJobPreview" function:
"Documentation Text Library" is the name of the print job.

**RPTJobPrint example**

```c
{
    //Print the specified print job out
    RPTJobPrint("@Text library (compact)");
}
```

Parameters of the RPTJobPrint function:
@Text library (compact) is the name of the print job.

**SysMalloc example**

```c
char* main(...);
{
    char* returnwert;
    char text[17];
    returnwert=SysMalloc(17);
    strcpy(returnwert,&text[0];
    return returnwert;
}
```
2.15.4.5 Examples - SetAlarmHigh to SetPropChar

SetAlarmHigh example

```c
{ //Set the upper limit for the warning
    SetAlarmHigh(lpszPictureName,"Bar1",3.0);
}
```

Parameters of the SetAlarmHigh function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Bar1" is the name of the object.
- "3.0" is the value to which the upper alarm limit will be set.

SetBackColor example

```c
{ //Set the back color blue
    SetBackColor(lpszPictureName,"StaticText1",CO_BLUE);
}
```

Parameters of the SetBackColor function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "StaticText1" is the name of the object.
- "CO_BLUE" is the constant for the color "Blue".

Note
Instead of using the constant for the color value you may also specify the color by means of a hexadecimal value.

SetBorderEndStyle example

```c
{ 
    SetBorderEndStyle(lpszPictureName,"Line", (2|393216));
}
```
Sets the left line end as filled arrow and the right one as filled circle. The left line end is stored in the two lower bytes, the right line end in the two upper bytes. The parameters are transferred by means of numeric values.

Example of setting the line ends with symbolic names

```c
{  
  SetBorderEndStyle(lpszPictureName,"Line", (LE_FULL_ARROW|LE_FULL_CIRCLE <<16));
}
```

Sets the left line end as filled arrow and the right one as filled circle. The left line end is stored in the two lower bytes, the right line end in the two upper bytes. To address the right line end the symbolic designation "LE_FULL_CIRCLE" is moved by 2 bytes or 16 bit into the two upper bytes.

SetBorderStyle example

```c
{  
  //Change the Border style  
  SetBorderStyle(lpszPictureName,"Rectangle1",3);
}
```

Parameters of the SetBorderStyle function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Rectangle1" is the name of the object.
"3" is the line style which is set for the object.

SetColorAlarmHigh example

```c
{  
  //Set the Color for the alarm high limit to red  
  SetColorAlarmHigh(lpszPictureName,"Bar1",CO_RED);
}
```

Parameters of the SetColorAlarmHigh function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.
"CO_RED" is the constant for the color red.

**Note**
Instead of using the constant for the color value you may also specify the color by means of a hexadecimal value.

### Example - SetCursorMode

```c
//Set the Cursor Mode to Alpha cursor
SetCursorMode(lpszPictureName,"GraphikObjekt1",FALSE);
```

Parameters of the SetCursorMode function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "GraphicObject1" is the name of the object.
- "FALSE" signifies: Cursor mode "Alpha-Cursor" is set.

### SetFilling example

```c
//Set the dynamic filling true
SetFilling(lpszPictureName,"Rectangle1",TRUE);
```

Parameters of the SetFilling function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Rectangle1" is the name of the object.
- "TRUE" means: Activating dynamic filling.

### SetFillingIndex example

```c
//Set the Filling of Rectangle1 to 10
SetFillingIndex(lpszPictureName,"Rectangle1",10);
```

Parameters of the SetFillingIndex function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Rectangle1" is the name of the object.
"10" is the fill level which is assigned to the object.

SetFillStyle example

{
    // Change the fill style
    SetFillStyle(lpszPictureName, "Rectangle1", 196617);
}

Parameters of the SetFillStyle function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Rectangle1" is the name of the object.
"196617" is the fill pattern (brick wall) which is set for the object.

SetFlashBackColor example

{
    // Set the flashing to True
    SetFlashBackColor(lpszPictureName, "Group1", TRUE);
}

Parameters of the SetFlashBackColor function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Group1" is the name of the object.
"TRUE" means: Activating flashing of the background color.

SetBackFlashColorOn example

{
    // Set the Flash color for the state on to red
    SetBackFlashColorOn(lpszPictureName, "Group1", CO_RED);
}

Parameters of the SetBackFlashColorOn function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Group1" is the name of the object.
"CO_Red" is the constant for the color "Red".

**Note**
Instead of using the constant for the color value you may also specify the color by means of a hexadecimal value.

**SetFlashRateFlashPic example**

```c
{ //Set the flash rate to 0
  SetFlashRateFlashPic(lpszPictureName,"Statusdisplay1",0);
}
```

Parameters of the SetFlashRateFlashPic function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Status display1" is the name of the object.
- "0" is the flash frequency of the object.

**SetFocus example**

```c
{ //Set the Focus on the Object Button 1
  Set_Focus(lpszPictureName,"Button1");
}
```

Parameters of the Set_Focus function
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Button1" is the name of the object on which the focus is set.

**SetFontBold example**

```c
{ //Set the displayed Text bold
  SetFontBold(lpszPictureName,"StatischerText1",TRUE);
}
```

Parameters of the SetFontBold function:
- "lpszPictureName" is the name of the picture in which the object was configured.
"StaticText1" is the name of the object.
"TRUE" means: The text is written in bold face.

SetFontSize example

{
//Set Font Size to 12
SetFontSize(lpszPictureName,"StaticText1",12);
}

Parameters of the SetFontSize function:
"lpszPictureName" is the name of the picture in which the object was configured.
"StaticText1" is the name of the object.
"12" is the font size to which the text is set.

SetHeight example

{
//Set the height of the object to 100
SetHeight(lpszPictureName,"WinCCLogo",100);
}

Parameters of the SetHeight function:
"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.
"100" is the height to which the object is set.

SetHiddenInput example

{
//Set the hidden input true
SetHiddenInput(lpszPictureName,"IOField1",TRUE);
}

Parameters of the SetHiddenInput function:
"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"TRUE" means: Activating the hidden input.

**SetLanguage example**

```c
//German
SetLanguage(0x0407);
```

The Runtime language is set to German.

**SetLeft example**

```c
//Set the x-position to 0
SetLeft(lpszPictureName,"WinCCLogo",0);
```

Parameters of the SetLeft function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "WinCCLogo" is the name of the object.
- "0" is the X position to which the object is set.

**SetLink example**

```c
LINKINFO linkinfo;

//Set the link type
linkinfo.LinkType = 1;

//Set the update cycle
linkinfo.dwCycle = 0;

//set the Structmember
strcpy(linkinfo.szLinkName, "U08i_link_00");

//Set the connection to the tag
SetLink(lpszPictureName,"Bar1","Process",&linkinfo);
```
Parameters of the SetLink function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.
"Process" is the property connected to a tag.
"&linkinfo" is the address of the linkinfo structure.

1. Set the connection type for the process property to direct connection.
2. Set the update cycle to "Upon change".
3. Set the tag name to U08i_link_00.

SetMarker example

```c
{
  //Set the marker visible
  SetMarker(lpszPictureName,"Bar1",TRUE);
}
```

Parameters of the SetMarker function:
"lpszPictureName" is the name of the picture in which the object was configured.
"Bar1" is the name of the object.
"TRUE" means: The marker is displayed.

SetOutputValueDouble example

```c
{
  //Set the output value of the IO field to 55.5
  SetOutputValueDouble(lpszPictureName,"IOField1",55.5);
}
```

Parameters of the SetOutputValueDouble function:
"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"55.5" is the value which is output.
SetPictureDown example

{
//Set the picture name to activated.bmp
SetPictureDown(lpszPictureName,"Roundbutton1","activated.bmp");
}

Parameters of the SetPictureDown function:
"lpszPictureName" is the name of the picture in which the object was configured.
"RoundButton1" is the name of the object.
"activated.bmp" is the picture name of the picture to be displayed in round button 1.

SetPictureName example

{
//Set the picture name cool_man.bmp
SetPictureName(lpszPictureName,"GraphicObject1","cool_man.bmp");
}

Parameters of the SetPictureName function:
"lpszPictureName" is the name of the picture in which the object was configured.
"GraphicObject1" is the name of the object.
"cool_man.bmp" is the picture name of the picture to be displayed in graphic object 1.

SetPictureUp example

{
//Set the picture name to deactivated.bmp
SetPictureUp(lpszPictureName,"Roundbutton1","deactivated.bmp");
}

Parameters of the SetPictureUp function:
"lpszPictureName" is the name of the picture in which the object was configured.
"RoundButton1" is the name of the object.
"deactivated.bmp" is the picture name of the picture to be displayed in round button 1.
SetPosition example

```c

//Set the Slider Position to 30
SetPosition(lpszPictureName,"Control1",30);
```

Parameters of the SetPosition function:

"lpszPictureName" is the name of the picture in which the object was configured.
"Control1" is the name of the object.
"30" is the position to which the slider is to be set.

SetPropBOOL example

```c

//Set the visibility TRUE
SetPropBOOL("lpszPictureName","EAFeld1","Visible",TRUE);
```

Parameters of the SetVisible function:

"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"TRUE" means: The object is intended to be visible.

SetPropChar example

```c

//Set the property Tooltiptext
SetPropChar("gs_graph_eafield","IOField1","ToolTipText","Tooltiptext1");
```

Parameters of the SetPropChar function:

"lpszPictureName" is the name of the picture in which the object was configured.
"IOField1" is the name of the object.
"Tooltiptext" is the object property.
"Tooltiptext 1" is the value to which the property is to be set.
2.15.4.6 Examples - SetRangeMax to SetWidth

SetRangeMax example

```c
{ //Set the Upper Scale Limit
    SetRangeMax(lpszPictureName,"Controll",80);
}
```

Parameters of the SetRangeMax function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Controll" is the name of the object.
- "80" is the upper limit to be assigned to the object.

SetRangeMin example

```c
{ //Set the lower Scale Limit
    SetRangeMin(lpszPictureName,"Controll",0);
}
```

Parameters of the SetRangeMin function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Controll" is the name of the object.
- "0" is the lower limit to be assigned to the object.

SetScaling example

```c
{ //Set the Scaling Visible
    SetScaling(lpszPictureName,"Bar1",TRUE);
}
```

Parameters of the SetScaling function:
- "lpszPictureName" is the name of the picture in which the object was configured.
- "Bar1" is the name of the object.
- "TRUE" means: Making the scaling visible.
SetTagBit example

```
{
    //Set the tag to true
    SetTagBit("gs_tag_bit", TRUE);
}
```

Parameters of the SetTagBit function:

- "gs_tag_bit" is the name of the tag.
- "TRUE" is the value to be written to the tag.

Beispiel SetTagBitStateWait

```
{
    DWORD dwstate;

    //Load dwState with default values
    dwstate = 0xFFFFFFFF;

    //Set the value of the tag to TRUE
    //dwstate is the tag state
    SetTagBitStateWait("gs_tag_bit", TRUE, &dwstate);

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the SetTagBitStateWait function:

- "gs_tag_bit" is the name of the tag.
- "TRUE" is the value to be written to the tag.
- "&dwstate" is the address of the tags in which the tag status is to be stored.

1. Setting the tags to the specified value.
2. Executing user-defined code for processing return values.

SetTagChar example

```
{
    //Set the tag to Example text
    SetTagChar("gs_tag_char", "Example Text");
}
```
Parameters of the SetTagChar function:
"gs_tag_char" is the name of the tag.
"Example text" is the value to be written to the tag.

SetTagCharStateWait example

{  
  DWORD dwstate;  
  
  //Load dwState with default values  
  dwstate = 0xFFFFFFFF;  
  
  //Set the tag to Example Text  
  //dwstate is the tag state  
  SetTagCharStateWait("gs_tag_char","Example Text",&dwstate);  
  
  //User defined code where the  
  //user can do something with the return value  
  ...  
}

Parameters of the SetTagCharStateWait function:
"gs_tag_char" is the name of the tag.
"Example text" is the value to be written to the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Setting the tags to the specified value.
2. Executing user-defined code for processing return values.

SetTagFloat example

{  
  //Set the tag to 55.4711  
  SetTagFloat("gs_tag_float",55.4711);  
}

Parameters of the SetTagFloat function:
"gs_tag_float" is the name of the tag.
"55.4711" is the value to be written to the tag.
SetTagFloatStateWait example

{
DWORD dwstate;
char szValue[9];

//Load dwState with default values
dwstate = 0xFFFFFFFF;

//Set the tag to 55.4711
//dwstate is the tag state
SetTagFloatStateWait("gs_tag_float",55.4711,&dwstate);

//User defined code where the
//user can do something with the return value
...
}

Parameters of the SetTagFloatStateWait function:
"gs_tag_float" is the name of the tag.
"55.4711" is the value to be written to the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Setting the tags to the specified value.
2. Executing user-defined code for processing return values.
SetTagMultiStateWait example

{
#define DATA_SIZE 5  
DWORD dwData[DATA_SIZE];

// define all tags
BOOL lValue1;
long lValue2;
char szValue3[_MAX_PATH];
float lValue4;
char lValue5;

// Fill the tags with the values
// you want to set into the WinCC tags
...

// Set the WinCC tags
SetTagMultiStateWait(&dwData, "%d%d%s%f%d", "gs_tag_bit", lValue1,  
"gs_tag_SByte", lValue2,  
"gs_tag_char", szValue3,  
"gs_tag_float", lValue4,  
"gs_tag_word", lValue5);

// User defined code where the  
// user can do something with the return value
...
}

Parameters of the SetTagMultiStateWait function:
"dwData" is the DWord-Array, in which the tag statuses are stored.
"%d%d%s%f%d" are the type descriptions of the tags to be written.
"gs_tag_bit" is the WinCC tag to be written.
"lValue1" is the tag to whose value the WinCC tag gs_tag_bit is to be set.
"gs_tag_SByte" is the WinCC tag to be written.
"lValue2" is the tag to whose value the WinCC tag gs_tag_SByte is to be set.
The other parameters are to be handled in the same way as those described previously.

1. Creating a DWord-Array with the required size (Number of tags).
2. Creating tags whose values are to be written to the WinCC tags.
3. Writing the values of the previously created and filled tags to the WinCC tags.
4. Executing user-defined code for processing return values.
SetTagMultiWait example

BOOL ok;

ok=SetTagMultiWait("%d%s%f", "Ernie_word", 16,
    "Ernie_char", "Hello World",
    "Ernie_double", 55.4711);

SetTagPrefix example

{
    //Set the TagPrefix to Struct1.
    SetTagPrefix(lpszPictureName,"PicWindow1","Struct1.");

    //Set the picture name again to update the tag prefix
    SetPictureName(lpszPictureName,"PicWindow1","gs_graph_eafield");
}

Parameters of the SetTagPrefix function:
"lpszPictureName" is the name of the picture in which the object was configured.
"PictureWindow1" is the name of the object.
"Struct1." is the tag prefix to be set at picture window 1.

1. Set the tag prefix of the object "PictureWindow1" to "Struct1."
2. Reset the name of the picture shown in the picture window to make the tag prefix setting effective.

SetTagRaw example

{
    #define DATA_SIZE 3

    BYTE byData[DATA_SIZE];

    // Fill the Byte array with the values
    // you want to set into the raw data tag
    ...

    //Set the tag to the default values
    SetTagRaw("gs_tag_raw", byData, DATA_SIZE);
}
Parameters of the SetTagRaw function:
"gs_tag_raw" is the name of the tag.
"byData" is the byte array whose values are written to the raw data tags.
"DATA_SIZE" is the number of values that will be written.

1. Creating a BYTE-Array with the required size (size of the raw data tag).
2. Filling the BYTE-Array with the values to be written.
3. Writing the values of the BYTE-Array to the raw data tag.

SetTagRawStateWait example

```c
{
#define DATA_SIZE 3

BYTE byData[DATA_SIZE];
DWORD dwstate;
char szValue[9];

//Load dwState with default values
dwstate = 0xFFFFFFFF;

// Fill the Byte array with the values
// you want to set into the raw data tag
...

//Set the tag to the default values
//dwstate is the tag state
SetTagRawStateWait("gs_tag_raw",byData,DATA_SIZE,&dwstate);

//User defined code where the
//user can do something with the return value
...
}
```

Parameters of the SetTagRawStateWait function:
"gs_tag_raw" is the name of the tag.
"byData" is the byte array whose values are written to the raw data tags.
"DATA_SIZE" is the number of values that will be written.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Creating a BYTE-Array with the required size (size of the raw data tag).
2. Filling the BYTE-Array with the values to be written.
3. Writing the values of the BYTE-Array to the raw data tag.
4. Executing user-defined code for processing return values.

**SetTagSByte example**

```c
{
    //Set the tag to 50
    SetTagSByte("gs_tag_SByte", 50);
}
```

Parameters of the `SetTagSByte` function:
- "gs_tag_SByte" is the name of the tag.
- "50" is the value to be written to the tag.

**Beispiel SetTagSByteStateWait**

```c
{
    DWORD dwstate;
    char szValue[9];

    //Load dwState with default values
    dwstate = 0xFFFFFFFF;

    //Set the tag to 50
    //dwstate is the tag state
    SetTagSByteStateWait("gs_tag_SByte", 50, &dwstate);

    //User defined code where the
    //user can do something with the return value
    ...
}
```

Parameters of the `SetTagSByteStateWait` function:
- "gs_tag_SByte" is the name of the tag.
- "50" is the value to be written to the tag.
- "&dwstate" is the address of the tags in which the tag status is to be stored.

1. Setting the tags to the specified value.
2. Executing user-defined code for processing return values.
SetTagWord example

{
  //Set the tag to 50
  SetTagWord("gs_tag_word",50);
}

Parameters of the SetTagWord function:
"gs_tag_word" is the name of the tag.
"50" is the value to be written to the tag.

Beispiel SetTagWordStateWait

{
  DWORD dwstate;
  char szValue[9];

  //Load dwState with default values
  dwstate = 0xFFFFFFFF;

  //Set the tag to 50
  //dwstate is the tag state
  SetTagWordStateWait("gs_tag_word",50,&dwstate);

  //User defined code where the
  //user can do something with the return value
  ...
}

Parameters of the SetTagWordStateWait function:
"gs_tag_word" is the name of the tag.
"50" is the value to be written to the tag.
"&dwstate" is the address of the tags in which the tag status is to be stored.

1. Setting the tags to the specified value.
2. Executing user-defined code for processing return values.
SetText example

```c
{
    //Set the text Example Text on the StaticText field
    SetText(lpszPictureName,"StaticText1","Example Text");
}
```

Parameters of the SetText function:

"lpszPictureName" is the name of the picture in which the object was configured.
"StaticText1" is the name of the object.
"ExampleText" is the text which is to be output.

SetTop example

```c
{
    //Set the y-position to 0
    SetTop(lpszPictureName,"WinCCLogo",140);
}
```

Parameters of the SetTop function:

"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.
"140" is the Y position to which the object is set.

SetVisible example

```c
{
    //Set the Object visible
    SetVisible(lpszPictureName,"GraphicObject1",TRUE);
}
```

Parameters of the SetVisible function:

"lpszPictureName" is the name of the picture in which the object was configured.
"GraphicObject1" is the name of the object.
"TRUE" means: The object is intended to be visible.
SetWidth example

{
    //Set the width of the object to 400
    SetWidth(lpszPictureName,"WinCCLogo",400);
}

Parameters of the SetWidth function
"lpszPictureName" is the name of the picture in which the object was configured.
"WinCCLogo" is the name of the object.
"400" is the width to which the object is set.

2.15.4.7 Examples of WinCC controls

How to add elements to a WinCC OnlineTrendControl

Introduction

In the following example, insert value columns with properties in an empty WinCC OnlineTableControl and link the columns to archive tags.

Prerequisite

- An archive is created in the "Tag Logging Editor" with three archive tags.
- A "WinCC OnlineTableControl" with the name "Control2" is inserted in the process picture in the Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured, for example, the event "mouse click" with a C action and the following script for the button.

```c
// Set the width of the object to 400
SetWidth(lpszPictureName,"WinCCLogo",400);
```
Example

//enable BackColor
SetPropBOOL(lpszPictureName, "Control2", "UseColumnBackColor", TRUE);

//add new TimeColumn and assign column length
SetPropChar(lpszPictureName, "Control2", "TimeColumnAdd", "myRefTimeColumn");
SetPropWord(lpszPictureName, "Control2", "TimeColumnLength", 20);

//add new ValueColumn and assign properties
SetPropChar(lpszPictureName, "Control2", "ValueColumnAdd", "myValueTable1");
SetPropWord(lpszPictureName, "Control2", "ValueColumnProvider", 1);
SetPropChar(lpszPictureName, "Control2", "ValueColumnTagName", "Process value archive\PDL_ZT_1");
SetPropWord(lpszPictureName, "Control2", "ValueColumnBackColor", RGB(255,255,255));
SetPropChar(lpszPictureName, "Control2", "ValueColumnTimeColumn", "myRefTimeColumn");

//add new ValueColumn and assign properties
SetPropChar(lpszPictureName, "Control2", "ValueColumnAdd", "myValueTable2");
SetPropWord(lpszPictureName, "Control2", "ValueColumnProvider", 1);
SetPropChar(lpszPictureName, "Control2", "ValueColumnTagName", "Process value archive\PDL_ZT_2");
SetPropWord(lpszPictureName, "Control2", "ValueColumnBackColor", RGB(0,255,255));
SetPropChar(lpszPictureName, "Control2", "ValueColumnTimeColumn", "myRefTimeColumn");

//add new ValueColumn and assign properties
SetPropChar(lpszPictureName, "Control2", "ValueColumnAdd", "myValueTable3");
SetPropWord(lpszPictureName, "Control2", "ValueColumnProvider", 1);
SetPropChar(lpszPictureName, "Control2", "ValueColumnTagName", "Process value archive\PDL_ZT_3");
SetPropWord(lpszPictureName, "Control2", "ValueColumnBackColor", RGB(255,255,0));
SetPropChar(lpszPictureName, "Control2", "ValueColumnTimeColumn", "myRefTimeColumn");

Result

![WinCC OnlineTableControl](image-url)
How to add elements to a WinCC OnlineTrendControl

Introduction

In the following example you insert the Trend Window, Value Axis, Time Axis and Trends elements into an empty WinCC OnlineTrendControl.

Prerequisite

- An archive is created in the "Tag Logging Editor" with three archive tags.
- A "WinCC OnlineTrendControl" with the name "Control2" is inserted in the process picture in the Graphics Designer.
- A button is inserted in the Graphics Designer. You have configured, for example, the event "mouse click" with a C action and the following script for the button.
Example

//create reference to new window, time and value axis
SetPropChar(lpszPictureName, "Control12", "TrendWindowAdd", "myWindow");
SetPropChar(lpszPictureName, "Control12", "TimeAxisAdd", "myTimeAxis");
SetPropChar(lpszPictureName, "Control12", "ValueAxisAdd", "myValueAxis");

//assign time and value axis to the window
SetPropChar(lpszPictureName, "Control12", "TimeAxisTrendWindow", "myWindow");
SetPropChar(lpszPictureName, "Control12", "ValueAxisTrendWindow", "myWindow");

//add new trend and assign properties
SetPropChar(lpszPictureName, "Control12", "TrendAdd", "myTrend1");
SetPropWord(lpszPictureName, "Control12", "TrendProvider", 1);
SetPropChar(lpszPictureName, "Control12", "TrendTagName", "Process value archive\PDL_ZT_1");
SetPropWord(lpszPictureName, "Control12", "TrendColor", RGB(255,0,0));
SetPropChar(lpszPictureName, "Control12", "TrendTrendWindow", "myWindow");
SetPropChar(lpszPictureName, "Control12", "TrendTimeAxis", "myTimeAxis");
SetPropChar(lpszPictureName, "Control12", "TrendValueAxis", "myValueAxis");

//add new trend and assign properties
SetPropChar(lpszPictureName, "Control12", "TrendAdd", "myTrend2");
SetPropWord(lpszPictureName, "Control12", "TrendProvider", 1);
SetPropChar(lpszPictureName, "Control12", "TrendTagName", "Process value archive\PDL_ZT_2");
SetPropWord(lpszPictureName, "Control12", "TrendColor", RGB(0,255,0));
SetPropChar(lpszPictureName, "Control12", "TrendTrendWindow", "myWindow");
SetPropChar(lpszPictureName, "Control12", "TrendTimeAxis", "myTimeAxis");
SetPropChar(lpszPictureName, "Control12", "TrendValueAxis", "myValueAxis");

//add new trend and assign properties
SetPropChar(lpszPictureName, "Control12", "TrendAdd", "myTrend3");
SetPropWord(lpszPictureName, "Control12", "TrendProvider", 1);
SetPropChar(lpszPictureName, "Control12", "TrendTagName", "Process value archive\PDL_ZT_3");
SetPropWord(lpszPictureName, "Control12", "TrendColor", RGB(0,0,255));
SetPropChar(lpszPictureName, "Control12", "TrendTrendWindow", "myWindow");
SetPropChar(lpszPictureName, "Control12", "TrendTimeAxis", "myTimeAxis");
SetPropChar(lpszPictureName, "Control12", "TrendValueAxis", "myValueAxis");

2.15.4.8 Examples - Obsolete functions

AXC_OnBtnMsgFirst example

{
    // jump to the first message in the WinCC Alarm Control
    A XC_OnBtnMsgFirst("gs_alarm_00", "Contro l1");
}
Parameters of the AXC_OnBtnMsgFirst function:
"gs_alarm_00" is the name of the picture in which WinCC Alarm Control was configured.
Control1 is the object name of the WinCC Alarm Control.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

Beispiel AXC_OnBtnMsgLast

```
{ // jump to the last message in the WinCC Alarm Control
  AXC_OnBtnMsgLast("gs_alarm_00","Control1");
}
```

Parameters of the AXC_OnBtnMsgLast function:
"gs_alarm_00" is the name of the picture in which WinCC Alarm Control was configured.
Control1 is the object name of the WinCC Alarm Control.

AXC_OnBtnScroll example

```
{ // activate/deactivate the scroll function
  AXC_OnBtnScroll("gs_alarm_00","Control1");
}
```

Parameters of the AXC_OnBtnScroll function:
"gs_alarm_00" is the name of the picture in which WinCC Alarm Control was configured.
Control1 is the object name of the WinCC Alarm Control.
AXC_OnBtnSinglAckn example

{
    // acknowledge the active message
    AXC_OnBtnSinglAckn("gs_alarm_00","Control1");
}

Parameters of the AXC_OnBtnSinglAckn function:

"gs_alarm_00" is the name of the picture in which WinCC Alarm Control was configured.
Control1 is the object name of the WinCC Alarm Control.

AXC_SetFilter example

{
    BOOL ret;
    MSG_FILTER_STRUCT Filter;
    CMN_ERROR Error;

    //Reset the filter struct
    memset( &Filter, 0, sizeof( MSG_FILTER_STRUCT ) );

    //Set the filter name
    strcpy( Filter.szFilterName, "Control1" );

    // Choose selection elements
    Filter.dwFilter = MSG_FILTER_NR_FROM | MSG_FILTER_NR_TO;

    // Message number from
    Filter.dwMsgNr[0] = 2;
    // Message number to
    Filter.dwMsgNr[1] = 2;

    ret = AXC_SetFilter("gs_alarm_00","Control1",&Filter,&Error);   
}

1. Name the filter.
2. Select the filter type.
3. Specify the filter criteria.
4. Set the filter.
Note
The filter type and the filter criteria are to be adapted, all other filter types are described in the filter structure.

ResetFilter example

```c
{  
BOOL ret;
MSG_FILTER_STRUCT Filter;
CMN_ERROR Error;

//delete the whole Filter struct
memset( &Filter, 0, sizeof( MSG_FILTER_STRUCT ) );

//set an empty filter struct
AXC_SetFilter("gs_alarm_00","Control1",&Filter,&Error);
}
```

1. Delete the filter structure.
2. Write empty values into the filter structure.

TlgGetNumberOfColumns example

```c
{  
char text[5];
long int columns

//get number of Columns
columns = GetNumberOfColumns("TableControl_01");

//convert long int to char
sprintf(text,"%d",columns);

//set text on TextField5
SetText(lpszPictureName,"StaticText5",text);
}
```

Parameters of the TlgGetNumberOfColumns function:
"TableControl_01" is the name of the WinCC Table Control.

1. Temporarily store the return value of the TlgGetNumberOfColumns in columns.
2. Temporarily store the return value in the text string.
3. Output the return value to a static text field.

---

**Note**

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

---

**TlgGetNumberOfRows example**

```c
{
    char text[5];
    long int rows;

    //get number of rows
    rows = TlgGetNumberOfRows("TableControl_01");

    //convert long int to char
    sprintf(text,"%d",rows);

    //set text on TextField5
    SetText(lpszPictureName,"StaticText5",text);
}
```

Parameters of the TlgGetNumberOfRows function:

TableControl_01 is the name of the WinCC Table Control.

1. Temporarily store the return value of the TlgGetNumberOfRows in rows.
2. Temporarily store the return value in the text string.
3. Output the return value to a static text field.
TlgGetRulerTimeTrend example

{
SYSTEMTIME systime;
WORD wHour;
WORD wMin;
WORD wSec;

char szTime[10];

//Get the current system time
systime = TlgGetRulerTimeTrend("TrendControl_01",0);

//Get the hour
wHour = systime.wHour;
//Get the minute
wMin = systime.wMinute;
//Get the second
wSec = systime.wSecond;

//sprintf(szTime,"%d:%d:%d",wHour,wMin,wSec);

//output the variable name
SetText(lpszPictureName,"StaticText7",szTime);
}

1. Read out the current system time.
2. Read out hour, minute and second from the SYSTEMTIME structure.
3. Create a string containing the time.
4. Output the current time.
TlgGetRulerVariableNameTrend example

```c
{ 
  char* pszVarName = NULL; 
  char szVarName[20]; 

  //Get the ruler variable name 
  pszVarName = TlgGetRulerVariableNameTrend("TrendControl_01",0); 

  if (pszVarName != NULL) 
  { 
    //Copy the string 
    strncpy(szVarName,pszVarName,19); 
  } 

  //output the variable name 
  SetText(lpszPictureName,"StaticText6",szVarName); 
}
```

Parameters of the TlgGetRulerVariableNameTrend function:

"TrendControl_01" is the name of the WinCC Trend Control.

"0" is the number of the trend.

1. Temporarily store the return value of the TlgGetRulerVariableNameTrend function in pszVarName.
2. If a valid value has been returned, copy the return value to szVarName.
3. Output the return value to a static text field.

Note

Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

TlgTrendWindowPressOpenDlgButton example

```c
{ 
  //Opens the Property Dialog 
  TlgTrendWindowPressOpenDlgButton("TrendControl_01"); 
}
```

Parameters of the TlgTrendWindowPressOpenDlgButton function:

"TrendControl_01" is the window title of WinCC Trend Control.
TlgTrendWindowPressStartStopButton example

{
    // start/stop the actualization
    TlgTrendWindowPressStartStopButton("TrendControl_01");
}

Parameters of the TlgTrendWindowPressStartStopButton function:
"TrendControl_01" is the window title of WinCC Trend Control.

Note
Various examples are offered for the function descriptions. For functions with a similar syntax, a selected function is used as a template in the example. This example must be adapted as well.

TlgTrendWindowPressZoomInButton example

{
    // zoom in
    TlgTrendWindowPressZoomInButton("TrendControl_01");
}

Parameters of the TlgTrendWindowPressZoomInButton function:
"TrendControl_01" is the window title of WinCC Trend Control.

TlgTrendWindowPressZoomOutButton example

{
    // zoom out
    TlgTrendWindowPressZoomOutButton("TrendControl_01");
}

Parameters of the TlgTrendWindowPressZoomOutButton function:
"TrendControl_01" is the window title of WinCC Trend Control.
2.15.5 Lists

2.15.5.1 Bar direction

<table>
<thead>
<tr>
<th>Bar direction</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>0</td>
</tr>
<tr>
<td>bottom</td>
<td>1</td>
</tr>
<tr>
<td>left</td>
<td>2</td>
</tr>
<tr>
<td>right</td>
<td>3</td>
</tr>
</tbody>
</table>

2.15.5.2 Bar Scaling

<table>
<thead>
<tr>
<th>Numeric value</th>
<th>Bar Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Linear (same weighting)</td>
</tr>
<tr>
<td>1</td>
<td>Logarithmic (low values emphasized)</td>
</tr>
<tr>
<td>2</td>
<td>Negative logarithmic (high values emphasized)</td>
</tr>
<tr>
<td>3</td>
<td>Automatic (linear)</td>
</tr>
<tr>
<td>4</td>
<td>Tangential (high and low values emphasized)</td>
</tr>
<tr>
<td>5</td>
<td>Square (high values emphasized)</td>
</tr>
<tr>
<td>6</td>
<td>Cubic (high values strongly emphasized)</td>
</tr>
</tbody>
</table>

2.15.5.3 Flash frequencies

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note

Since the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time etc.).

The information in the table is therefore only for orientation purposes.
2.15.5.4 I/O field, output format

The display of numeric values output into an I/O field is controlled by a format specification.

A format indication consists of one or several formatting characters. The valid formatting characters and their meaning are listed in the following table:

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
<th>Note</th>
</tr>
</thead>
</table>
| s         | Positive numbers are displayed with signs    | Always in the first position of the format specification  
            |                                               | May only appear once in the format specification |
| 0(ZERO)   | Leading and ending zeros are output.         | Always following s                             |
|           |                                               | If s is missing, it is in the first position   |
|           |                                               | May only appear once in the format specification |
| 9         | Specifies the position of a digit in the number to be output | May appear in the format indication as often as required. |
| ,(comma)  | Position of the decimal point                |                                                |
| e         | Returns the number in exponential form       | Always at the last position of the format specification |

Example:

<table>
<thead>
<tr>
<th>Number</th>
<th>Format</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>123,4</td>
<td>999,999</td>
<td>123,455</td>
</tr>
<tr>
<td>123,4</td>
<td>999,99</td>
<td>123,46</td>
</tr>
<tr>
<td>123,4</td>
<td>9999,9999</td>
<td>123,4550</td>
</tr>
<tr>
<td>123,4</td>
<td>s09999.999</td>
<td>+0123,4550</td>
</tr>
<tr>
<td>123,4</td>
<td>9.999999e</td>
<td>1.23455e+002</td>
</tr>
</tbody>
</table>

If the decimal point is left out in the format specification the decimal places are not displayed and the number is rounded to an integer.

If fewer decimal positions are provided in the format specification than the number actually has, only the decimal places specified in the format specification are output.

The number is rounded correspondingly.

If the number has more places before the decimal point than specified in the format specification, three asterisks (***') are output which means that the number cannot be displayed in this format.
2.15.5.5 I/O field, data type of the field content

<table>
<thead>
<tr>
<th>Data type</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>0</td>
</tr>
<tr>
<td>decimal</td>
<td>1</td>
</tr>
<tr>
<td>string</td>
<td>2</td>
</tr>
<tr>
<td>hexadecimal</td>
<td>3</td>
</tr>
</tbody>
</table>

2.15.5.6 I/O field, field type

<table>
<thead>
<tr>
<th>Type</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition</td>
<td>0</td>
</tr>
<tr>
<td>Input</td>
<td>1</td>
</tr>
<tr>
<td>Output and input</td>
<td>2</td>
</tr>
</tbody>
</table>

2.15.5.7 Element alignment in check boxes and radio boxes

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>-1</td>
</tr>
</tbody>
</table>

2.15.5.8 Color chart

The 16 primary colors are:

<table>
<thead>
<tr>
<th>Color</th>
<th>Color value (Hex)</th>
<th>Symbolic constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>0x000000FF</td>
<td>CO_RED</td>
</tr>
<tr>
<td>Dark red</td>
<td>0x00000080</td>
<td>CO_DKRED</td>
</tr>
<tr>
<td>Green</td>
<td>0x0000FF00</td>
<td>CO_GREEN</td>
</tr>
<tr>
<td>Dark green</td>
<td>0x00008000</td>
<td>CO_DKGREEN</td>
</tr>
<tr>
<td>Blue</td>
<td>0x00FF0000</td>
<td>CO_BLUE</td>
</tr>
<tr>
<td>Color</td>
<td>Color value (Hex)</td>
<td>symbolic constant</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Dark blue</td>
<td>0x00800000</td>
<td>CO_DKBLUE</td>
</tr>
<tr>
<td>Cyan</td>
<td>0x00FFFF00</td>
<td>CO_CYAN</td>
</tr>
<tr>
<td>Dark cyan</td>
<td>0x00808000</td>
<td>CO_DKCYAN</td>
</tr>
<tr>
<td>Yellow</td>
<td>0x0000FFFF</td>
<td>CO_YELLOW</td>
</tr>
<tr>
<td>Dark yellow</td>
<td>0x00008080</td>
<td>CO_DKYELLOW</td>
</tr>
<tr>
<td>Magenta</td>
<td>0x00FF00FF</td>
<td>CO_MAGENTA</td>
</tr>
<tr>
<td>Dark magenta</td>
<td>0x00800080</td>
<td>CO_DKMAGENTA</td>
</tr>
<tr>
<td>Light gray</td>
<td>0x00C0C0C0</td>
<td>CO_LTGRAY</td>
</tr>
<tr>
<td>Gray</td>
<td>0x00808080</td>
<td>CO_DKGRAY</td>
</tr>
<tr>
<td>Black</td>
<td>0x00000000</td>
<td>CO_BLACK</td>
</tr>
<tr>
<td>White</td>
<td>0x00FFFFF</td>
<td>CO_WHITE</td>
</tr>
</tbody>
</table>

**Note**

The symbolic constants are externally predefined by `#define` and provided by WinCC.

### 2.15.5.9 Format descriptors

For format descriptors the following type is expected:

- `%d = DWORD / Int`
- `%f = double`
- `%s = char*`

It is also possible e.g. to read a text tag with `%d` if provisions are made that the value can be mapped in a DWORD.

The following provision makes sure the value can be mapped:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>C-tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>Byte</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>SByte</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>Word</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>SWord</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>DWord</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
<tr>
<td>SDWord</td>
<td>%d</td>
<td>DWORD / long int signed</td>
</tr>
</tbody>
</table>
### Variable Format C-tag

<table>
<thead>
<tr>
<th>Variable Format</th>
<th>C-tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td>%f</td>
</tr>
<tr>
<td>Double</td>
<td>%f</td>
</tr>
<tr>
<td>Char</td>
<td>%s</td>
</tr>
</tbody>
</table>

**Note**

If a "DWORD," for which the 32nd bit is set, is to be read, a format descriptor must be used for unsigned integers (%u).

### 2.15.5.10 Fill pattern

<table>
<thead>
<tr>
<th>Fill pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>65536</td>
</tr>
<tr>
<td>Solid</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fill pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1048576</td>
<td>196611</td>
</tr>
<tr>
<td>1048577</td>
<td>196612</td>
</tr>
<tr>
<td>1048578</td>
<td>196613</td>
</tr>
<tr>
<td>1048579</td>
<td>196614</td>
</tr>
<tr>
<td>1048832</td>
<td>196615</td>
</tr>
<tr>
<td>1048833</td>
<td>196616</td>
</tr>
<tr>
<td>1048834</td>
<td>196617</td>
</tr>
<tr>
<td>1048835</td>
<td>196618</td>
</tr>
<tr>
<td>131072</td>
<td>196619</td>
</tr>
<tr>
<td>131073</td>
<td>196620</td>
</tr>
<tr>
<td>131073</td>
<td>196620</td>
</tr>
</tbody>
</table>
### 2.15.5.11 Line styles

<table>
<thead>
<tr>
<th>Line style</th>
<th>symbolic name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>__________</td>
<td>LS_SOLID</td>
<td>0</td>
</tr>
<tr>
<td>__________</td>
<td>LS_DASH</td>
<td>1</td>
</tr>
<tr>
<td>__________</td>
<td>LS_DOT</td>
<td>2</td>
</tr>
<tr>
<td>__________</td>
<td>LS_DASHDOT</td>
<td>3</td>
</tr>
<tr>
<td>__________</td>
<td>LS_DASHDOT-DOT</td>
<td>4</td>
</tr>
<tr>
<td>hidden</td>
<td>LS_INVISIBLE</td>
<td>5</td>
</tr>
</tbody>
</table>

### 2.15.5.12 Line end style

<table>
<thead>
<tr>
<th>Line end</th>
<th>symbolic name</th>
<th>Value for the left line ends</th>
<th>Value for the right line ends</th>
</tr>
</thead>
<tbody>
<tr>
<td>__________</td>
<td>LE_NO</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>→</td>
<td>LE_HOLLOW_ARROW</td>
<td>1</td>
<td>65536</td>
</tr>
<tr>
<td>⊲</td>
<td>LE_FULL_ARROW</td>
<td>2</td>
<td>131072</td>
</tr>
</tbody>
</table>

Note

The "Solid" fill pattern fills the object with the set background color.
### 2.15.5.13 List types

<table>
<thead>
<tr>
<th>List type</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>0</td>
</tr>
<tr>
<td>Binary</td>
<td>1</td>
</tr>
<tr>
<td>bit</td>
<td>2</td>
</tr>
</tbody>
</table>

### 2.15.5.14 Language ID

WinCC only supports the SUBLANG_DEFAULT languages of Windows.

<table>
<thead>
<tr>
<th>symbolic name</th>
<th>Value (hexadecimal)</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG_ARABIC</td>
<td>0x0401</td>
<td></td>
</tr>
<tr>
<td>LANG_AFRIKAANS</td>
<td>0x0436</td>
<td></td>
</tr>
<tr>
<td>LANG_ALBANIAN</td>
<td>0x041C</td>
<td></td>
</tr>
<tr>
<td>LANG_BASQUE</td>
<td>0x042D</td>
<td></td>
</tr>
<tr>
<td>LANG_BULGARIAN</td>
<td>0x0402</td>
<td></td>
</tr>
<tr>
<td>LANG_BYELORUSSIAN</td>
<td>0x0423</td>
<td></td>
</tr>
<tr>
<td>LANG_CATALAN</td>
<td>0x0403</td>
<td></td>
</tr>
<tr>
<td>LANG_CHINESE</td>
<td>0x0404</td>
<td></td>
</tr>
<tr>
<td>LANG_CROATIAN</td>
<td>0x041A</td>
<td></td>
</tr>
<tr>
<td>LANG_CZECH</td>
<td>0x0405</td>
<td>CSY</td>
</tr>
<tr>
<td>LANG_DANISH</td>
<td>0x0406</td>
<td>DAN</td>
</tr>
<tr>
<td>LANG_DUTCH</td>
<td>0x0413</td>
<td>NLD</td>
</tr>
<tr>
<td>LANG_ENGLISH</td>
<td>0x0409</td>
<td>ENU</td>
</tr>
<tr>
<td>LANG_ESTONIAN</td>
<td>0x0425</td>
<td></td>
</tr>
<tr>
<td>LANG_FAEROESE</td>
<td>0x0438</td>
<td></td>
</tr>
<tr>
<td>LANG_FARSI</td>
<td>0x0429</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

From a line width > 5 the line end "empty circle" is displayed as filled circle.
<table>
<thead>
<tr>
<th>symbolic name</th>
<th>Value (hexadecimal)</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG_FINNISH</td>
<td>0x040B</td>
<td>FIN</td>
</tr>
<tr>
<td>LANG_FRENCH</td>
<td>0x040C</td>
<td>FRA</td>
</tr>
<tr>
<td>LANG_GERMAN</td>
<td>0x0407</td>
<td>DEU</td>
</tr>
<tr>
<td>LANG_GREEK</td>
<td>0x0408</td>
<td></td>
</tr>
<tr>
<td>LANG_HEBREW</td>
<td>0x040D</td>
<td></td>
</tr>
<tr>
<td>LANG_HUNGARIAN</td>
<td>0x040E</td>
<td>HUN</td>
</tr>
<tr>
<td>LANG_ICELANDIC</td>
<td>0x040F</td>
<td>ISL</td>
</tr>
<tr>
<td>LANG_INDONESIAN</td>
<td>0x0421</td>
<td></td>
</tr>
<tr>
<td>LANG_ITALIAN</td>
<td>0x0410</td>
<td>ITA</td>
</tr>
<tr>
<td>LANG_JAPANESE</td>
<td>0x0411</td>
<td></td>
</tr>
<tr>
<td>LANG_KOREAN</td>
<td>0x0412</td>
<td></td>
</tr>
<tr>
<td>LANG_LATVIAN</td>
<td>0x0426</td>
<td></td>
</tr>
<tr>
<td>LANG_LITHUANIAN</td>
<td>0x0427</td>
<td></td>
</tr>
<tr>
<td>LANG_NORWEGIAN</td>
<td>0x0414</td>
<td>NOR</td>
</tr>
<tr>
<td>LANG_POLISH</td>
<td>0x0415</td>
<td>PLK</td>
</tr>
<tr>
<td>LANG_PORTUGUESE</td>
<td>0x0416</td>
<td>PTB</td>
</tr>
<tr>
<td>LANG_ROMANIAN</td>
<td>0x0418</td>
<td></td>
</tr>
<tr>
<td>LANG RUSSIAN</td>
<td>0x0419</td>
<td>RUS</td>
</tr>
<tr>
<td>LANG_SLOVAK</td>
<td>0x041B</td>
<td>SKY</td>
</tr>
<tr>
<td>LANG_SLOVENIAN</td>
<td>0x0424</td>
<td></td>
</tr>
<tr>
<td>LANG_SORBIA</td>
<td>0x042E</td>
<td></td>
</tr>
<tr>
<td>LANG SPANISH</td>
<td>0x040A</td>
<td>ESP</td>
</tr>
<tr>
<td>LANG SWEDISH</td>
<td>0x041D</td>
<td>SVE</td>
</tr>
<tr>
<td>LANG THAI</td>
<td>0x041E</td>
<td></td>
</tr>
<tr>
<td>LANG TURKISH</td>
<td>0x041F</td>
<td>TRK</td>
</tr>
<tr>
<td>LANG UKRAINIAN</td>
<td>0x0422</td>
<td></td>
</tr>
</tbody>
</table>

2.15.5.15 Text alignment

<table>
<thead>
<tr>
<th>Horizontal</th>
<th>Alignment</th>
<th>Numeric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>centered</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>right</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
### 2.15.5.16 Tag statuses

<table>
<thead>
<tr>
<th>Value (decimal)</th>
<th>Value (hexadecimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>0x0001</td>
<td>Connection to partner not established</td>
</tr>
<tr>
<td>2</td>
<td>0x0002</td>
<td>Handshake error</td>
</tr>
<tr>
<td>4</td>
<td>0x0004</td>
<td>Network module defective</td>
</tr>
<tr>
<td>8</td>
<td>0x0008</td>
<td>Configured upper limit exceeded</td>
</tr>
<tr>
<td>16</td>
<td>0x0010</td>
<td>Configured lower limit exceeded</td>
</tr>
<tr>
<td>32</td>
<td>0x0020</td>
<td>Format upper limit exceeded</td>
</tr>
<tr>
<td>64</td>
<td>0x0040</td>
<td>Format lower limit exceeded</td>
</tr>
<tr>
<td>128</td>
<td>0x0080</td>
<td>Conversion error</td>
</tr>
<tr>
<td>256</td>
<td>0x0100</td>
<td>Tag initialization value</td>
</tr>
<tr>
<td>512</td>
<td>0x0200</td>
<td>Tag replacement value</td>
</tr>
<tr>
<td>1024</td>
<td>0x0400</td>
<td>Channel addressing error</td>
</tr>
<tr>
<td>2048</td>
<td>0x0800</td>
<td>Tag not found or not available</td>
</tr>
<tr>
<td>4096</td>
<td>0x1000</td>
<td>Access to tag not permitted</td>
</tr>
<tr>
<td>8192</td>
<td>0x2000</td>
<td>Timeout, no check-back message from the channel</td>
</tr>
<tr>
<td>16384</td>
<td>0x4000</td>
<td>Server not available.</td>
</tr>
</tbody>
</table>
2.15.6 Structure definitions

2.15.6.1 Structure definition CCAPErrorExecute

typedef struct {
    DWORD dwCurrentThreadID;   // Thread ID of the current thread
    DWORD dwErrorCode1;       // Error code 1
    DWORD dwErrorCode2;       // Error code 2
    BOOL bCycle;              // cycle/acycle
    char* szApplicationName;  // Name of the application
    char* szFunctionName;     // Name of the function
    char* szTagName;          // Name of the tag
    LPVOID lpParam;           // Pointer to the action stack
    DWORD dwParamSize;        // Size of the action stack
    DWORD dwCycle;            // Cycle of the variable
    CMN_ERROR* pError;        // Pointer to CMN_ERROR
} CCAPErrorExecute;

Members

The meaning of the individual error IDs and the structure elements depending on them are specified in the following table:

<table>
<thead>
<tr>
<th>dwErrorCode1</th>
<th>dwErrorCode2</th>
<th>bCycle</th>
<th>szApplicationName</th>
<th>szFunctionName</th>
<th>szTagName</th>
<th>lpParam</th>
<th>dwParamSize</th>
<th>dwCycle</th>
<th>pError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1007001</td>
<td>0</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Action requires exception</td>
</tr>
<tr>
<td>1007001</td>
<td>1</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Exception when accessing the return result</td>
</tr>
<tr>
<td>1007001</td>
<td>4097</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Stack overflow while executing the action</td>
</tr>
<tr>
<td>1007001</td>
<td>4098</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>The action contains a division by 0</td>
</tr>
<tr>
<td>1007001</td>
<td>4099</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>The action contains an access to a non-existing symbol</td>
</tr>
<tr>
<td>1007001</td>
<td>4100</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>The action contains an access violation</td>
</tr>
<tr>
<td>1007004</td>
<td>0</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Function is not known</td>
</tr>
</tbody>
</table>
### Error structure

The OnErrorExecute function uses the error structure to evaluate or to output error messages, if marked by an "x" in the pError column.

### See also

Structure definition CMN_ERROR (Page 1601)

### 2.15.6.2 Structure definition CCAPTime

```c
typedef struct {
    DWORD dwCurrentThreadID; // ThreadID of the current Thread
    DWORD dwCode;            // Code
    BOOL bCycle;             // cycle/acycle
    char* szApplicationName; // Name of the Application
    char* szFunctionName;    // Name of the Function
    LPVOID lpParam;          // Pointer to the Action-Stack
    DWORD dwParamSize;       // size of the Action-Stack
    double dblTime;
    DWORD dwFlags;           // flags
} CCAPTime;
```
Members

dwCode
The structure element dwCode provides information on calling OnTime:

<table>
<thead>
<tr>
<th>dwCode</th>
<th>Call with time definition for each action</th>
<th>dwCode = 113</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwCode = 114</td>
<td>Call with time monitoring for each action</td>
<td></td>
</tr>
</tbody>
</table>


dwFlags
The structure element dwFlags provides information on the output type:

<table>
<thead>
<tr>
<th>dwFlags</th>
<th>The results are output to a file</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwFlags = TRUE</td>
<td>The results are output to a file</td>
</tr>
<tr>
<td>dwFlags = FALSE</td>
<td>The results are output to the diagnostics window</td>
</tr>
</tbody>
</table>

2.15.6.3 Structure definition CMN_ERROR

```c
struct CMNERRORSTRUCT {
    DWORD dwError1,
    DWORD dwError2,
    DWORD dwError3,
    DWORD dwError4,
    DWORD dwError5;
    TCHAR szErrorText[MAX_ERROR_LEN];
};
```

Description
The extended error structure contains the error code and an error text for the error that has occurred. Each application can use the error structure to evaluate or to output error messages.

Members

dwError1 .. dwError5
These entries can be used in any way by the API functions.
The API descriptions inform about the values the respective entries contain in case of an error.
If not specified otherwise, the error codes are present in dwError1.
2.15.6.4 Structure definition DM_TYPEREF

```c
typedef struct {
    DWORD dwType;
    DWORD dwSize;
    char szTypeName[MAX_DM_TYPE_NAME + 1];
} DM_TYPEREF;
```

**Members**

**dwType**
Specifies the tag type

<table>
<thead>
<tr>
<th>DM_VARTYPE_BIT</th>
<th>Binary tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM_VARTYPE_SBYTE</td>
<td>Signed 8-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_BYTE</td>
<td>Unsigned 8-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_SWORD</td>
<td>Signed 16-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_WORD</td>
<td>Unsigned 16-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_SDWORD</td>
<td>Signed 32-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_DWORD</td>
<td>Unsigned 32-bit value</td>
</tr>
<tr>
<td>DM_VARTYPE_FLOAT</td>
<td>Floating-point number 32-bit</td>
</tr>
<tr>
<td>DM_VARTYPE_DOUBLE</td>
<td>Floating-point number 64-bit</td>
</tr>
<tr>
<td>DM_VARTYPE_TEXT_8</td>
<td>Text tag, 8-bit font</td>
</tr>
<tr>
<td>DM_VARTYPE_TEXT_16</td>
<td>Text tag, 16-bit font</td>
</tr>
<tr>
<td>DM_VARTYPE_RAW</td>
<td>Raw data type</td>
</tr>
<tr>
<td>DM_VARTYPE_STRUCT</td>
<td>Structure tag</td>
</tr>
<tr>
<td>DM_VARTYPE_TEXTREF</td>
<td>Text reference tag</td>
</tr>
</tbody>
</table>

**dwSize**
Specifies the length of the data type in bytes.

**szTypeName**
In the case of structure tags, contains the name of the structure type.

szErrorText
Buffer for the text description of the error cause
The content is determined from the resources and therefore language-dependent.
2.15.6.5 Structure definition DM_VAR_UPDATE_STRUCT

typedef struct {
    DM_TYPEREF dmTypeRef;
    DM_VARKEY dmVarKey;
    VARIANT dmValue;
    DWORD dwState;
} DM_VAR_UPDATE_STRUCT;

Members

dmTypeRef
Contains information on the tag type. For performance reasons, nothing is entered into this structure in case of cyclic requirements.

dmVarKey
Specifies the tags to be edited.

dmValue
Tag value
Upon access to the value of the VARIANT a ".u." has to be inserted between the name of the VARIANT and the name of the member.

Example:

// Supply variant
myVariant.vt = VT_I4;
myVariant.u.lVal = 233;

A description of the data type VARIANT can be found in the associated documentation. The VARIANT dmValue must be initialized with VariantInit() before first use and enabled again with VariantClear(&dmValue) after use. For this reason, the structure DM_VAR_UPDATE_STRUCT must not be deleted with ZeroMemory() or memset().

dwState
Identifies the tag status.

See also

Tag statuses (Page 1598)
Structure definition DM_VARKEY (Page 1605)
Structure definition DM_TYPEREF (Page 1602)
2.15.6.6 Structure definition DM_VAR_UPDATE_STRUCTEX

```c
typedef struct {
    DM_TYPEREF dmTypeRef;
    DM_VARKEY dmVarKey;
    VARIANT dmValue;
    DWORD dwState;
    DWORD dwQualityCode;
} DM_VAR_UPDATE_STRUCTEX;
```

**Members**

- **dmTypeRef**
  Contains information on the tag type. For performance reasons, nothing is entered into this structure in case of cyclic requirements.

- **dmVarKey**
  Specifies the tags to be edited.

- **dmValue**
  Tag value

  Upon access to the value of the VARIANT a ".u." has to be inserted between the name of the VARIANT and the name of the member.

  ```c
  Example:
  // Supply variant
  myVariant.vt = VT_I4;
  myVariant.u.lVal = 233;
  ```

  A description of the data type VARIANT can be found in the associated documentation. The VARIANT dmValue must be initialized with VariantInit() before first use and enabled again with VariantClear(&dmValue) after use. For this reason, the structure DM_VAR_UPDATE_STRUCTEX must not be deleted with ZeroMemory() or memset().

- **dwState**
  Identifies the tag status.

- **dwQualityCode**
  Identifies the tag quality code.
See also

Tag statuses (Page 1598)
Structure definition DM_VARKEY (Page 1605)
Structure definition DM_TYPEREF (Page 1602)

2.15.6.7 Structure definition DM_VARKEY

typedef struct {
    DWORD dwKeyType;
    DWORD dwID;
    char szName[MAX_DM_VAR_NAME + 1];
    LPVOID lpvUserData;
} DM_VARKEY;

Members

dwKeyType
Defines whether the tag is to be addressed by a key ID or by its name.
DM_VARKEY_ID Specification via key ID
DM_VARKEY_NAME Specification via tag name

dwID
Contains the key ID of the tags if dwKeyType is set accordingly

szName
Contains the name of the tag if dwKeyType is set accordingly

lpvUserData
Pointer to application-specific data

2.15.6.8 Structure definition LINKINFO

typedef struct {
    LINKTYPE LinkType;
    DWORD dwCycle;
    TCHAR szLinkName[256];
} LINKINFO;
ANSI-C for Creating Functions and Actions

2.15 ANSI-C function descriptions

Members

LinkType

LinkType are enumeration constants defined in the "Trigger.h" file. They are to be integrated into your script with the #include "Trigger.h" command and the corresponding enumeration constants.

<table>
<thead>
<tr>
<th>LinkType</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUBRT_LT_NOLINK</td>
<td>0</td>
<td>no shortcut</td>
</tr>
<tr>
<td>BUBRT_LT_VARIABLE_DIRECT</td>
<td>1</td>
<td>direct tag</td>
</tr>
<tr>
<td>BUBRT_LT_VARIABLE_INDIRECT</td>
<td>2</td>
<td>indirect tag</td>
</tr>
<tr>
<td>BUBRT_LT_ACTION</td>
<td>3</td>
<td>C action</td>
</tr>
<tr>
<td>BUBRT_LT_ACTION_WIZARD</td>
<td>4</td>
<td>Dynamic Dialog</td>
</tr>
<tr>
<td>BUB_LT_DIRECT_CONNECTION</td>
<td>5</td>
<td>Direct connection</td>
</tr>
<tr>
<td>BUBRT_LT_ACTION_WIZARD_INPROC</td>
<td>6</td>
<td>Dynamic Dialog</td>
</tr>
</tbody>
</table>

For the function SetLink only the enumeration constants BUBRT_LT_VARIABLE_DIRECT and BUBRT_LT_VARIABLE_INDIRECT may be used. The function GetLink allows to return all listed enumeration constants.

dwCycle

Update cycle time

<table>
<thead>
<tr>
<th>dwCycle</th>
<th>Update Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>Picture cycle</td>
</tr>
<tr>
<td>235</td>
<td>Window Cycle</td>
</tr>
<tr>
<td>0</td>
<td>Upon change</td>
</tr>
<tr>
<td>1</td>
<td>250ms</td>
</tr>
<tr>
<td>2</td>
<td>500 ms</td>
</tr>
<tr>
<td>3</td>
<td>1 s</td>
</tr>
<tr>
<td>4</td>
<td>2 s</td>
</tr>
<tr>
<td>5</td>
<td>5s</td>
</tr>
<tr>
<td>6</td>
<td>10s</td>
</tr>
<tr>
<td>7</td>
<td>1min</td>
</tr>
<tr>
<td>8</td>
<td>5min</td>
</tr>
<tr>
<td>9</td>
<td>10min</td>
</tr>
<tr>
<td>10</td>
<td>1h</td>
</tr>
<tr>
<td>11-15</td>
<td>User cycle 1-5</td>
</tr>
</tbody>
</table>

szLinkName

Tag name
### 2.15.6.9 Structure definition MSG_FILTER_STRUCT

```c
typedef struct {
    CHAR        szFilterName[MSG_MAX_TEXTLEN+1];
    WORD        dwFilter;
    SYSTEMTIME  st[2];
    DWORD       dwMsgNr[2];
    DWORD       dwMsgClass;
    DWORD       dwMsgType[MSG_MAX_CLASS];
    DWORD       dwMsgState;
    WORD        wAGNr[2];
    WORD        wAGSubNr[2];
    DWORD       dwArchiMod;
    char        szTB[MSG_MAX_TB][MSG_MAX_TBCONTENT+1];
    DWORD       dwTB;
    Double      dPValue[MSG_MAX_PVALUE][2];
    DWORD       dwPValue[2];
    DWORD       dwMsgCounter[2];
    DWORD       dwQuickSelect;
} MSG_FILTER_STRUCT;
```

### Description
In this structure the filter criteria are specified.

### Members

**dwFilter**

The filter conditions are defined by means of the following constants from the file "m_global.h":

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG_FILTER_DATE_FROM</td>
<td>Date from</td>
</tr>
<tr>
<td>MSG_FILTER_DATE_TO</td>
<td>Date to</td>
</tr>
<tr>
<td>MSG_FILTER_TIME_FROM</td>
<td>Time from</td>
</tr>
<tr>
<td>MSG_FILTER_TIME_TO</td>
<td>Time to</td>
</tr>
<tr>
<td>MSG_FILTER_NR_FROM</td>
<td>Message number from</td>
</tr>
<tr>
<td>MSG_FILTER_NR_TO</td>
<td>Message number to</td>
</tr>
<tr>
<td>MSG_FILTER_CLASS</td>
<td>Message classes</td>
</tr>
<tr>
<td>MSG_FILTER_STATE</td>
<td>Message status</td>
</tr>
<tr>
<td>MSG_FILTER_AG_FROM</td>
<td>AS number from</td>
</tr>
<tr>
<td>MSG_FILTER_AG_TO</td>
<td>AS number to</td>
</tr>
<tr>
<td>MSG_FILTER_AGSUB_FROM</td>
<td>AG subnumber from</td>
</tr>
<tr>
<td>MSG_FILTER_AGSUB_TO</td>
<td>AG subnumber to</td>
</tr>
<tr>
<td>MSG_FILTER_TEXT</td>
<td>Message texts</td>
</tr>
<tr>
<td>MSG_FILTER_PVALUE</td>
<td>Process values</td>
</tr>
</tbody>
</table>
MSG_FILTER_COUNTER_FROM | Internal message counter from
MSG_FILTER_COUNTER_TO | Internal message counter to
MSG_FILTER_PROCESSMSG | Process messages
MSG_FILTER_SYSMSG | System messages
MSG_FILTER_BEDMSG | Operator messages
MSG_FILTER_DATE | Date from to
MSG_FILTER_TIME | Time from to
MSG_FILTER_NR | Message number from to
MSG_FILTER_VISIBLEONLY | Display visible messages
MSG_FILTER_HIDDENONLY | Display hidden messages

st
Date/time from - to
Where st[0] is the start time (from), st[1] the end time (to)
Assign these fields for the filter criteria: MSG_FILTER_DATE, MSG_FILTER_DATE_FROM,
MSG_FILTER_DATE_TO, MSG_FILTER_TIME, MSG_FILTER_TIME_FROM or
MSG_FILTER_TIME_TO
If a current time is needed for the transfer of a SYSTEMTIME parameter the function
GetLocalTime is to be used instead of GetSystemTime. As a rule there is a significant time
difference between these two functions.

dwMsgNr
Message number from - to
Where dwMsgNr[0] is the start no. (from), dwMsgNr[1] the end no. (to)
Assign these fields for the filter criteria: MSG_FILTER_NR, MSG_FILTER_NR_FROM or
MSG_FILTER_NR_TO

dwMsgClass
Message classes bit-coded.
Assign this field for the filter criterion: MSG_FILTER_CLASS

dwMsgType
Message type per message class, bit-coded
Assign this field for the filter criterion: MSG_FILTER_CLASS

dwMsgState
Message status bit-coded.
Assign this field for the filter criterion: MSG_FILTER_STATE

wAGNr
AGNr from - to
Assign these fields for the filter criteria: MSG_FILTER_AG_FROM or MSG_FILTER_AG_TO
wAGSubNr
AGSubNr from - to
Assign this field for the filter criteria: MSG_FILTER_AGSUB_FROM or
MSG_FILTER_AGSUB_TO

dwArchivMode
Archiving / logging
Must be assigned 0.

szTB
Texts of the text blocks
Assign these fields for the filter criterion: MSG_FILTER_TEXT

dwTB
Active text blocks (from - to, bit-coded)
Assign this field for the filter criterion: MSG_FILTER_TEXT

dPValue
Process values from - to
Assign these fields for the filter criterion: MSG_FILTER_PVALUE

dwPValue
Active process values (from - to, bit-coded)
Assign this field for the filter criterion: MSG_FILTER_PVALUE

dwMsgCounter
Internal message counter from - to
Assign these fields for the filter criteria: MSG_FILTER_COUNTER_FROM,
MSG_FILTER_COUNTER_TO

dwQuickSelect
Quick selection for hour, day, month
The parameter is reserved for future upgrades and must be preset to 0.
Assign this field for the filter criterion: MSG_FILTER_QUICKSELECT
LOWORD type:

<table>
<thead>
<tr>
<th>MSG_FILTER_QUICK_MONTH</th>
<th>Quick selection last n months</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG_FILTER_QUICK_DAYS</td>
<td>Quick selection last n days</td>
</tr>
<tr>
<td>MSG_FILTER_QUICK_HOUR</td>
<td>Quick selection last n hours</td>
</tr>
</tbody>
</table>

HIWORD number: 1...n
The end time of the quick selection refers to the current system time of the local computer.
The start time is calculated back n * ( months, days, hours ).
2.15.6.10 Structure definition MSG_RTDATA_STRUCT

typedef struct {
    DWORD     dwMsgState;
    DWORD     dwMsgNr;
    SYSTEMTIME stMsgTime;
    DWORD     dwTimeDiff;
    DWORD     dwCounter;
    DWORD     dwFlags;
    WORD      wPValueUsed;
    WORD      wTextValueUsed;
    double    dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL_STRUCT mtTextValue[MSG_MAX_PVALUE];
} MSG_RTDATA_STRUCT;

Members

dwMsgState
Message status

<table>
<thead>
<tr>
<th>Enum</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG_STATE_COME</td>
<td>0x00000001</td>
<td>Message came in</td>
</tr>
<tr>
<td>MSG_STATE_GO</td>
<td>0x00000002</td>
<td>Message went out</td>
</tr>
<tr>
<td>MSG_STATE_QUIT</td>
<td>0x00000003</td>
<td>Message acknowledged</td>
</tr>
<tr>
<td>MSG_STATE_LOCK</td>
<td>0x00000004</td>
<td>Message locked</td>
</tr>
<tr>
<td>MSG_STATE_UNLOCK</td>
<td>0x00000005</td>
<td>Message unlocked</td>
</tr>
<tr>
<td>MSG_STATE_QUIT_SYSTEM</td>
<td>0x00000010</td>
<td>Message acknowledged by system</td>
</tr>
<tr>
<td>MSG_STATE_QUIT_EMERGENCY</td>
<td>0x00000011</td>
<td>Emergency acknowledgement</td>
</tr>
<tr>
<td>MSG_STATE_QUIT_HORN</td>
<td>0x00000012</td>
<td>Horn acknowledgement</td>
</tr>
<tr>
<td>MSG_STATE_COMEGO</td>
<td>0x00000013</td>
<td>Message came in and went out, only display in message list</td>
</tr>
<tr>
<td>MSG_STATE_UPDATE</td>
<td>0x00010000</td>
<td>Bit for message update</td>
</tr>
<tr>
<td>MSG_STATE_RESET</td>
<td>0x00020000</td>
<td>Bit for message reset</td>
</tr>
<tr>
<td>MSG_STATE_SUMTIME</td>
<td>0x00040000</td>
<td>Bit active for daylight savings time</td>
</tr>
<tr>
<td>MSG_STATE_INSTANCE</td>
<td>0x00080000</td>
<td>Bit for instance message (n messages of a no.)</td>
</tr>
</tbody>
</table>

dwMsgNr
Message number

stMsgTime
Date/Time: Telegram time depending on the calling function

dwTimeDiff
Duration coming/Telegram time in seconds

dwCounter
Internal message counter
**dwFlags**

Message flags in the database

<table>
<thead>
<tr>
<th>dwFlag</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG_FLAG_SUMTIME</td>
<td>0x00000001</td>
<td>Daylight savings time active</td>
</tr>
<tr>
<td>MSG_FLAG_COMMENT</td>
<td>0x00000002</td>
<td>Message has comments</td>
</tr>
<tr>
<td>MSG_FLAG_ARCHIV</td>
<td>0x00000004</td>
<td>Archiving</td>
</tr>
<tr>
<td>MSG_FLAG_PROTOCOL</td>
<td>0x00000008</td>
<td>Logging</td>
</tr>
<tr>
<td>MSG_FLAG_TEXTVALUES</td>
<td>0x00000010</td>
<td>Message has values accompanying the text</td>
</tr>
<tr>
<td>MSG_FLAG_TIMEINVALID</td>
<td>0x00000020</td>
<td>Bit for invalid date/time stamp</td>
</tr>
<tr>
<td>MSG_FLAG_INSTANCE</td>
<td>0x00000040</td>
<td>Instance message identification (185269)</td>
</tr>
</tbody>
</table>

**wPValueUsed**

Process values used, bit-coded. Every bit may only be set in one of the two structure elements "wPValueUsed" or "wTextValueUsed". An accompanying value may either be a number or a text.

**wTextValueUsed**

text values used, bit-coded. Every bit may only be set in one of the two structure elements "wPValueUsed" or "wTextValueUsed". An accompanying value may either be a number or a text.
2.15 ANSI-C function descriptions
3.1 Automated configuration

Contents

You can use VBA to automate configuration in Graphics Designer. This comprises:

- Adaptation of the Graphics Designer
- Editing of pictures
- Editing of objects
- Dynamizing with VBA
- Access to external applications

A VBA editor is available for this purpose in the "Graphics Designer" editor.

This chapter contains

- a brief introduction on how to use VBA in WinCC,
- basic information on using VBA in Graphics Designer and
- reference to the VBA object model in Graphics Designer.
3.2 Introduction: Using VBA in WinCC

3.2.1 Introduction: Using VBA in WinCC

Introduction
You have a VBA editor available in Graphics Designer which allows you to configure pictures automatically. The VBA editor is identical to the one from the products in the Microsoft Office family. You can make direct use of your VBA programming experience.

Principle
With VBA you extend the functionality of the Graphics Designer and automate configuring. You may use VBA in Graphics Designer as follows, including:

- Creating user-defined menus and toolbars
- Creating and editing Standard, Smart and Windows objects
- Adding dynamics to properties of pictures and objects
- Configuring actions in pictures and objects
- Accessing products that support VBA (e.g. products in the MS Office family)

A description of the VBA object model for the Graphics Designer is provided in this documentation in the section under "VBA Reference".

See also
Organizing VBA Code in a WinCC Project (Page 1615)
VBA Reference (Page 1745)
VBA in the Graphics Designer (Page 1621)
Executing VBA Macros in Graphics Designer (Page 1619)
Differentiation: Using VBA (Page 1614)

3.2.2 Differentiation: Using VBA

Introduction
You can use VBA exclusively for configuration and functional enhancement in the Graphics Designer. The following indicates where there are better options available for efficient configuration or where VBA cannot be used.
VB and C Scripts

VB and C scripts are active only at runtime and are used to dynamize picture and object properties as well as in action configuration.

Dynamic Wizards

The dynamic wizards are not replaced by VBA. However, VBA allows you to enhance the functionality of the dynamic wizards with ease.

ODK

ODK comprises function calls that enable access to all the functionality of WinCC both in the configuration system and in runtime. In contrast to ODK, VBA offers simple object-oriented access to the objects of the Graphics Designer.

See also

VBA in the Graphics Designer (Page 1621)
Executing VBA Macros in Graphics Designer (Page 1619)
Organizing VBA Code in a WinCC Project (Page 1615)
Introduction: Using VBA in WinCC (Page 1614)

3.2.3 Organizing VBA Code in a WinCC Project

Introduction

You organize the VBA code for your WinCC project in the VBA editor. This is where you specify whether the VBA code is to be available in only one picture, in the entire project or in all projects. Depending on where you place the VBA code, the term used to refer to the code is:

- global VBA code,
- project-specific VBA code or
- picture-specific VBA code.

Note

A picture in the Graphics Designer is known as a "document" in the VBA object model.

The VBA editor

To start the VBA editor in the Graphics Designer, press <ALT+F11> or choose "Tools" > "Macros" > "Visual Basic Editor". If you have not yet opened a picture in the Graphics Designer, you can only edit the global or project-specific VBA code.

The global and project-specific data and all open pictures are displayed in the VBA editor's Project Explorer.
Global VBA code (1)

Refers to VBA code that you write to the "GlobalTemplateDocument" in the VBA editor. This VBA code is saved in the "@GLOBAL.PDT" file, which is located in the WinCC installation directory.

The VBA code that you put in the "GlobalTemplateDocument" is the code that you want to be made available in all WinCC projects on your computer. If you need the VBA code on a different computer, use the export and import functions in the VBA editor.

A WinCC computer uses only the @GLOBAL.PDT stored locally in the WinCC installation directory (…\Siemens\WinCC\Templates).

Note

When you perform an update installation, your global "@Global.pdt" template is saved in the "@Global.sav" backup file. The backup file is saved in the …\Siemens\WinCC\Templates directory. Your VBA code from the old global template is not automatically applied to the new global template.

Applying the VBA Code from the Old Global Template:
In order to apply the VBA code from the old template after an update installation, proceed as follows:

1. If you have already entered VBA code into the new global template, open the VBA editor in the Graphics Designer and copy the VBA code.
2. Close WinCC.
3. Open the ...\Siemens\WinCC\Templates directory in Windows Explorer.
4. Delete the new global template "@Global.pdt".
5. Rename the "@Global.sav" backup file to "@Global.pdt".
6. If you have already copied VBA code from the new global template, open the VBA editor in the Graphics Designer and insert the copied VBA code.

The VBA code from your old global template is available again.

Project-specific VBA code (2)

Refers to VBA code that you write to the "ProjectTemplateDocument" in the VBA editor. This VBA code is saved in the "@PROJECT.PDT" file, which is located in the root directory of each WinCC project.

The "@PROJECT.PDT" file has a reference to the "@GLOBAL.PDT" file. Functions and procedures which you have saved in the "@GLOBAL.PDT" file can be called up directly in the "ProjectTemplateDocument".

The "ProjectTemplateDocument" is where you put VBA code that you want to use in all pictures in the open project. If you need the VBA code on a different computer, use the export and import functions in the VBA editor.

You can open and edit the "@PROJECT.PDT" file in the same way as a PDL file. This will allow you to use the "@PROJECT.PDT" file as a template: For example, you may create there the basic picture of your system which will then be automatically transferred into each new PDL file of the project. Picture properties such as layers or zoom are not copied to the PDL file, nor is the VBA code.

Picture-specific VBA code (3)

Refers to VBA code that you write to the document "This Document" relating to the corresponding picture in the VBA editor. This VBA code is saved as a PDL file together with the picture.

The PDL file has a reference to the "@PROJECT.PDT" file. Functions and procedures which you have saved in the "@PROJECT.PDT" file can be called up directly from the PDL file. However, you do not have access to functions or procedures that are stored in the "@GLOBAL.PDT" file.

Note

You can create modules, class modules and user forms in each document.

You can protect the VBA code of a module against unauthorized access by setting a password. To do this, select the "Tools" > "VBAObject Properties" menu item in the VBA editor.
Special features during the execution of VBA macros

For the execution of VBA macros, the following applies: Initially picture-specific VBA code is executed, followed by project-specific VBA code. If therefore you call a VBA macro that is contained for example both in the picture and in the project-specific VBA code, only the VBA macro from the picture is executed. This has the effect of preventing VBA macros and functions from being executed twice, which otherwise can lead to errors.

In connection with event handling the forwarding of events is activated by default. You can prevent events from being forwarded if you want to respond to an event in the picture-specific VBA code only.

Additional information on this topic is given under "Event Handling".

Testing with the Debugger

You can test your VB scripts at runtime with the VBA editor's debugger. You may find additional information in the help system of the VBA editor.

See also

Event Handling (Page 1709)
VBA in the Graphics Designer (Page 1621)
Executing VBA Macros in Graphics Designer (Page 1619)
How to export and import VBA code (Page 1618)

3.2.4 How to export and import VBA code

Principle

In the VBA editor you can import and export VBA code, enabling you to transfer it to another computer. References to procedures and functions which you call within the project are therefore retained.

Note

When you import VBA code you must enter references to external libraries manually after the import process, on the target computer.

Procedure

Exporting VBA code

1. In the VBA editor's Project Explorer, select the module, whose VBA code you want to export.
2. Choose the "File" > "Export File menu command".
3. Select the path and enter the file name.
4. Click "Save".

The VBA code is exported to a file. The file type depends on the module from which the VBA code was exported.

**Importing VBA code**
1. In the VBA editor's Project Explorer, select the document into which you want to import the VBA code.
2. Choose the menu option "File" > "Import File".
3. Select the file and click "Open" in order to import the VBA code as "ThisDocument" into the "Class Modules" folder.
4. In the "Class Modules" folder, open the document "ThisDocument" and copy the VBA code into the document in the required project.

**See also**
Organizing VBA Code in a WinCC Project (Page 1615)

### 3.2.5 Executing VBA Macros in Graphics Designer

**Introduction**

Three possibilities are available to you for executing VBA macros in the Graphics Designer:
- Event Handling
- User-defined menu or toolbar
- VBA editor

**Event Handling**

Predefined events (such as the opening of a picture) can occur in the Graphics Designer, the active picture or the component library, to which you can respond with VBA event handlers. These events occur only during configuring in the Graphics Designer and have nothing to do with the events of action configuring.

In this example, a brief message is to be issued when a picture is opened. The "Opened event" is used for this:

```vba
Private Sub Document_Opened(CancelForwarding As Boolean)
    MsgBox ("Picture was opened!")
End Sub
```

Further information on the subject of event handling is provided under "Event handling" and "Events".
User-defined menu or toolbar

VBA allows you to create user-defined menus and toolbars in the Graphics Designer. You can assign a VBA macro to each user-defined menu entry or icon; this macro is then executed when you click on the menu entry or the icon. This way you can extend the functionality of the Graphics Designer to suit your requirements.

Further information on the creation of user-defined menus and toolbars is provided under "Creating your own menus and toolbars".

VBA editor

You can start a VBA macro in the VBA editor by pressing <F5>. If you press <F8>, you can execute a VBA macro step by step.

See also

VBA Reference (Page 1745)
Event Handling (Page 1709)
Creating Customized Menus and Toolbars (Page 1627)
VBA in the Graphics Designer (Page 1621)
Organizing VBA Code in a WinCC Project (Page 1615)
Introduction: Using VBA in WinCC (Page 1614)
3.3 VBA in the Graphics Designer

3.3.1 VBA in the Graphics Designer

Introduction
You use VBA in the Graphics Designer in order to automate frequently recurring steps during configuring. You can create user-defined menus and toolbars in order to make it easier to execute the VBA macros that you have created.

Basically, in the Graphics Designer you can replace all configuring work that you would otherwise perform with the mouse with VBA macros. This applies in particular to the GUI (layers and zoom) and the editing of objects in pictures including dynamics.

Adapting the Graphics Designer with VBA
The Graphics Designer is represented by the Application object in VBA. With VBA you can carry out configuring in the Graphics Designer in several languages, create user-defined menus and toolbars and access the component library.

Editing Pictures with VBA
A picture in the Graphics Designer is represented by the Document object.

With VBA you can access the properties of the picture and edit settings for layers and the zoom factors. As well as this you can create picture-specific menus and toolbars. These are only visible, however, for as long as the picture is active.

Editing Objects with VBA
An object in the picture is represented by the HMIObject object. With VBA you can create and delete objects and access the object properties. For example, with VBA you can very quickly create a large number of objects with identical properties for your plant display.

Creating Dynamics with VBA
VBA enables you to add dynamics to properties and events of pictures and objects.

Event Handling
With VBA you can respond to events that occur in the Graphics Designer or in a picture, for example when you insert a new object into a picture. You use event handling in order to execute VBA macros in certain program situations.
Access to external applications

You can use VBA to access programs which support VBA, for example products in the Microsoft Office family. This means that you have an opportunity to read values from an Excel table and then assign them to object properties.

Note

Access to applications that were compiled with .net

You need to recompile applications that were compiled with .net to enable access to VBA in Graphics Designer.

See also

Editing Pictures with VBA (Page 1654)
SymbolLibrary Object (Page 2074)
HMIOBJECT Object (Page 1988)
Document Object (Page 1949)
Application Object (Page 1909)
Accessing External Applications with VBA (Page 1713)
Event Handling (Page 1709)
Creating Dynamics with VBA (Page 1686)
Editing Objects with VBA (Page 1660)
Adapting the Graphics Designer with VBA (Page 1623)
Introduction: Using VBA in WinCC (Page 1614)
3.3.2 Adapting the Graphics Designer with VBA

3.3.2.1 Adapting the Graphics Designer with VBA

Introduction

In VBA the Application object represents the Graphics Designer:

Access to the component library

VBA gives you full access to the component library. You can extend the component library with VBA by for example creating and deleting folders or copying objects and inserting them into a picture.

User-defined menus and toolbars

You can create user-defined menus and toolbars in order to execute VBA macros in the Graphics Designer. In this way you can extend the functionality of the Graphics Designer to suit your particular requirements.

Language-dependent configuring

With VBA you can carry out configuring in the Graphics Designer in more than one language. You therefore have access to the language-dependent object properties and you can create the user-defined menus and toolbars in different languages.

See also

Editing Pictures with VBA (Page 1654)
Accessing the component library with VBA (Page 1646)
Creating Customized Menus and Toolbars (Page 1627)
Language-Dependent Configuration with VBA (Page 1624)
VBA in the Graphics Designer (Page 1621)
3.3.2.2 Language-Dependent Configuration with VBA

Introduction

With VBA you can carry out configuring in the Graphics Designer for several different languages. This gives you access to the language-dependent properties of objects in the Graphics Designer, while you can also make the user-defined menus and toolbars available in different languages. In VBA, foreign-language texts are stored in a list of the "LanguageTexts" type. The settings for language-dependent fonts are stored in a list of the "LanguageFonts" type.

Further information about language-dependent configuring is also provided in the WinCC documentation "Setting up multilingual projects".

User interface language

You can only switch to a different desktop language in WinCC, not with VBA. When you switch desktop language in WinCC, the "DesktopLanguageChanged" event is triggered. You can adapt the user-defined menus and toolbars to suit the user, for example, by replacing language-dependent tool icons.

The following objects and the associated language-dependent properties react to the switching of the user interface language:

- FolderItem Object
- Menu object and MenuItem object
- ToolbarItem Object

Further information about the desktop language is provided in the WinCC documentation "Setting up multilingual projects" under "Language terms in WinCC".

As of WinCC V7.3, VBA is installed in all languages that you have selected in the WinCC setup. When you open the "VBA" editor in the Graphics Designer, the "VBA" editor opens in the same user interface language as the Graphics Designer.

Project language

You can change the configuring language with VBA using the "CurrentDataLanguage" property.

In this example the configuring language is changed to "English":

Sub ChangeCurrentDataLanguage()
  'VBA1
  Application.CurrentDataLanguage = 1033
  MsgBox "The Data language has been changed to english"
  Application.CurrentDataLanguage = 1031
  MsgBox "The Data language has been changed to german"
End Sub

All language-dependent properties, such as ToolTipText, are affected by the change.
Configuring for more than one language in VBA

There are two possible ways for you to carry out configuring for several languages with VBA.

- **Language switching**: Text properties of objects.
- **Text language lists**: Text properties of user-defined menus and toolbars, and objects.

**Language change**

You can change the language-dependent properties (e.g., "Text") of objects with VBA. To do this, assign the text to the corresponding property and then change the configuring language in order to assign the text in the other language.

**LanguageTexts listing**

You can save the multilingual texts for the respective object directly in the associated listing of the "LanguageTexts" type. To do this, enter the language ID for the language and the associated text.

The list of language codes is available in the WinCC documentation (Index > Language Code).

In this example a German label and an English label are assigned to the button "myButton":

```vba
Sub AddLanguagesToButton()
    'VBA
    Dim objLabelText As HMILanguageText
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    'Set default label:
    objButton.Text = "Default-Text"
    'Add English label:
    Set objLabelText = objButton.LDTexts.Add(1033, "English Text")
    'Add German label:
    Set objLabelText = objButton.LDTexts.Add(1031, "German Text")
End Sub
```

**See also**

- LanguageTexts Object (Listing) (Page 2000)
- LanguageFonts Object (Listing) (Page 1997)
- How to assign help texts to menus and toolbars (Page 1639)
- How to create menus in multiple languages (Page 1633)
- VBA Reference (Page 1745)
3.3.2.3 Creating Customized Menus and Toolbars

Configuring Menus and Toolbars

Introduction

You can "liven up" user-defined menus and toolbars so that they respond to certain program situations in the Graphics Designer. For example, if an icon is not available because no object is selected, you can gray out the icon. A check mark before a menu item can indicate, for example, whether a selection is activated.

The following illustration shows you the configuration possibilities, using the example of a user-defined menu:

```
'VBA13
```

Active (yes/no) (1)
Activates the entry or dims it. You can use the "Enabled" property for user-defined menus, menu items and icons:

```
'VBA14
```

Marked with check mark (yes/no) (2)
Marks the menu item with a check mark. You can only use the "Checked" property for user-defined menu items:

```
'VBA15
```

Shortcut (3)
Defines a key combination for a menu item or an icon. You can only use the "Shortcut" property for user-defined menu items and icons:
Visible (yes/no) (4)
Displays or hides the item. You can use the "Visible" property for user-defined menus, menu items and toolbars and for their icons:

'VBA16

Creating Customized Menus and Toolbars

User-defined menus and toolbars in the Graphics Designer
You can use the following user-defined menus and toolbars in the Graphics Designer to run VBA macros:

- User-defined menus and toolbars are always visible when the Graphics Designer is open. You should use application-specific menus and toolbars when the VBA macros that are to be executed from them have to be accessible at all times.
- Picture-specific menus and toolbars are linked to a specific picture and remain visible for as long as the picture is active. You should use picture-specific menus and toolbars when the VBA macros used there are relevant only for that particular picture.
Positioning of user-defined menus and toolbars

In the case of user-defined menus, the "Position" parameter determines the final positioning in the menu bar:

- Application-specific menus (1) are always positioned to the right of the "Windows" menu in the Graphics Designer, while picture-specific menus (2) are always positioned to the left of the "Help" menu in the Graphics Designer.

However, application-specific toolbars are not treated as "preferred". In this case, the positioning is determined by the order in which you insert the toolbars. Toolbars are positioned below the Graphics Designer toolbar.

Properties of user-defined menus and toolbars

In the case of user-defined menus and toolbars you can use hyphens to divide entries, for example according to certain categories. As well as this you can also create submenus in a user-defined menu.

The following configuration options are available to you for user-defined menus and toolbars and their entries:

- Visible (yes/no): Displays or hides the item (visible property).
- Active (yes/no): Activates the entry or dims it (enabled property).
- Marked with check mark (yes/no) - only available for menu command (Checked property).
- Shortcut: Key combination for calling a menu command (ShortCut property).
• Statustext: Text that is displayed in the status bar (StatusText property).
• Tooltip text - only available for an icon (ToolTipText property).

You can hide a menu command, for example, if the macro cannot be executed at a certain time. In this way you can prevent inadvertent wrong operation.

You can create all texts and labels of user-defined menus and toolbars in multiple languages so that the user-defined menus and toolbars can also react to a language change.

See also

How to assign VBA macros to menus and toolbars (Page 1642)
How to assign help texts to menus and toolbars (Page 1639)
How to Add a New Icon to the Toolbar (Page 1637)
How to Create an Application-specific Toolbar (Page 1635)
How to create menus in multiple languages (Page 1633)
How to add a new menu entry to a menu (Page 1631)
How to Create Picture-specific Menus and Toolbars (Page 1655)
How to Create a New Application-Specific Menu (Page 1629)
Configuring Menus and Toolbars (Page 1626)
Executing VBA Macros in Graphics Designer (Page 1619)

How to Create a New Application-Specific Menu

Introduction

Application-specific menus remain visible even when all pictures in Graphics Designer are closed. You can use the Started event, for example, in order to insert an application-specific menu at an early stage.

Position the VBA code either

• in the "GlobalTemplateDocument" if you want the menu to be available in all projects, or
• in the "ProjectTemplateDocument" if you want the menu to be available in the current project.
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:

   ![Project Explorer Screenshot]

3. To create a user-defined menu in the Graphics Designer, you can for example insert a "CreateApplicationMenus()" procedure in the document. In this example, two user-defined menus are created:

   ```vba
   Sub CreateApplicationMenus()
   'Declaration of menus...:
   Dim objMenu1 As HMIMenu
   Dim objMenu2 As HMIMenu
   'Add menus. Parameters are "Position", "Key" und "DefaultLabel":
   Set objMenu1 = Application.CustomMenus.InsertMenu(1, "AppMenu1", "App_Menu_1")
   End Sub
   ```

4. Start the procedure with <F5>.

Result

The two menus "App_Menu_1" and "App_Menu_2" are inserted to the right of the "Window" menu:

![Menu Screenshot]

See also

Creating Customized Menus and Toolbars (Page 1627)

InsertMenu Method (Page 1850)

How to assign VBA macros to menus and toolbars (Page 1642)

How to assign help texts to menus and toolbars (Page 1639)
How to add a new menu entry to a menu

Requirements
You must have created the user-defined menu first.

Introduction
You can insert three different types of menu items in the user-defined menu:

- **Menu entry**: To call VBA macros.
- **Separator line**: For clearer design of user-defined menu.
- **Submenu**: Same as user-defined menu (e.g. command structuring).

The "Position" parameter determines the order of the menu items within the user-defined menu.

The "Key" parameter is a unique identification of the menu item. This parameter is used if you use the "MenuItemClicked" event for calling VBA macros.

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:
3. To create menu items in a previously created user-defined menu, you can for example insert an "InsertMenuItems()" procedure in the document. In this example a number of menu items are created in the user-defined menu "App_Menu_1":

```
Sub InsertMenuItems()
'VBA4
Dim objMenu1 As HMIMenu
Dim objMenu2 As HMIMenu
Dim objMenuItem1 As HMIMenuItem
Dim objSubMenu1 As HMIMenuItem
'Create Menu:
Set objMenu1 = Application.CustomMenus.InsertMenu(1, "AppMenu1", "App_Menu_1")
'Next lines add menu-items to userdefined menu.
'Parameters are "Position", "Key" and DefaultLabel:
Set objMenuItem1 = objMenu1.MenuItems.InsertMenuItem(1, "mItem1_1", "App_MenuItem_1")
Set objMenuItem1 = objMenu1.MenuItems.InsertMenuItem(2, "mItem1_2", "App_MenuItem_2")
'Adds seperator to menu ("Position", "Key")
Set objMenuItem1 = objMenu1.MenuItems.InsertSeparator(3, "mItem1_3")
'Adds a submenu into a userdefined menu
Set objSubMenu1 = objMenu1.MenuItems.InsertSubMenu(4, "mItem1_4", "App_SubMenu_1")
'Adds a menu-item into a submenu
Set objMenuItem1 = objSubMenu1.SubMenu.InsertMenuItem(5, "mItem1_5", "App_SubMenuItem_1")
Set objMenuItem1 = objSubMenu1.SubMenu.InsertMenuItem(6, "mItem1_6", "App_SubMenuItem_2")
End Sub
```

4. Start the procedure with <F5>.

Result

The "InsertMenuItems()" procedure inserts the menu "App_Menu_1" with these menu items:
How to create menus in multiple languages

Introduction
You can create a user-defined menu that responds to a change of language. To do this you need to define the necessary number of labels in other languages for the menu and for each menu item.

The foreign-language label comprises the language ID (LCID) and the foreign-language text (DisplayName).

The list of language codes is available in the WinCC documentation (Index > Language Code).

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:
3. To define multilingual labels for a user-defined menu, you can for example insert a "MultipleLanguagesForAppMenu1()" procedure in the document. In this example English labels are defined for the "App_Menu_1" menu:

```vba
Sub InsertMenuItems()
'VBA5
'Execute this procedure first
Dim objMenu1 As HMIMenu
Dim objMenu2 As HMIMenu
Dim objMenuItem1 As HMIMenuItem
Dim objSubMenu1 As HMIMenuItem
'Insert Menu:
Set objMenu1 = Application.CustomMenus.InsertMenu(1, "AppMenu1", "App_Menu_1")
'Next lines inserts menu-items to userdefined menu.
'parameters are "Position", "Key" and DefaultLabel:
Set objMenuItem1 = objMenu1.MenuItems.InsertMenuItem(1, "mItem1_1", "App_MenuItem_1")
Set objMenuItem1 = objMenu1.MenuItems.InsertMenuItem(2, "mItem1_2", "App_MenuItem_2")
'Inserts seperator into menu ("Position", "Key")
Set objMenuItem1 = objMenu1.MenuItems.InsertSeparator(3, "mItem1_3")
'Inserts a submenu into a userdefined menu
Set objSubMenu1 = objMenu1.MenuItems.InsertSubMenu(4, "mItem1_4", "App_SubMenu_1")
' Inserts a menu-item into a submenu
Set objMenuItem1 = objSubMenu1.SubMenu.InsertMenuItem(5, "mItem1_5", "App_SubMenuItem_1")
Set objMenuItem1 = objSubMenu1.SubMenu.InsertMenuItem(6, "mItem1_6", "App_SubMenuItem_2")
End Sub
Sub MultipleLanguagesForAppMenu1()
'execute this procedure after "InsertMenuItems()" was run
'Object "objLanguageTextMenu1" contains the foreign-language labels for the menu
Dim objLanguageTextMenu1 As HMILanguageText
'Object "objLanguageTextMenu1Item" contains the foreign-language labels for the menu-items
Dim objLanguageTextMenu1Item As HMILanguageText
Dim objMenu1 As HMIMenu
Dim objSubMenu1 As HMIMenuItem
Set objMenu1 = Application.CustomMenus("AppMenu1")
Set objSubMenu1 = Application.CustomMenus("AppMenu1").MenuItems("mItem1_4")
'Inserts foreign-language label into a menu:
'("Add(LCID, DisplayName)" method:
```
Set objLanguageTextMenu1 = objMenu1.LDLabelTexts.Add(1033, "English_App_Menu_1")

' Inserts foreign-language label into a menuitem:
Set objLanguageTextMenuItem1 = objMenu1.MenuItems("mItem1_1").LDLabelTexts.Add(1033, "My first menu item")

' Adds a foreign-language label into a submenu:
Set objLanguageTextMenuItem1 = objSubMenu1.SubMenu.Item("mItem1_5").LDLabelTexts.Add(1033, "My first submenu item")
End Sub

4. Start the procedure with <F5>.

Result

If you now switch the configuring language to English, certain items in the user-defined menu are shown in English.

See also

LanguageTexts Object (Listing) (Page 2000)
LDLabelTexts Property (Page 2299)
How to assign VBA macros to menus and toolbars (Page 1642)
How to assign help texts to menus and toolbars (Page 1639)
How to add a new menu entry to a menu (Page 1631)
How to Create a New Application-Specific Menu (Page 1629)
Configuring Menus and Toolbars (Page 1626)
Creating Customized Menus and Toolbars (Page 1627)
Language-Dependent Configuration with VBA (Page 1624)

How to Create an Application-specific Toolbar

Introduction

Application-specific toolbars remain visible even when all pictures in the Graphics Designer are closed.

Position the VBA code either

- in the "GlobalTemplateDocument" if you want the toolbar to be available in all projects, or
- in the "ProjectTemplateDocument" if you want the toolbar to be available in the current project.
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:

3. To create a user-defined toolbar in the Graphics Designer, you can for example insert a "CreateApplicationToolbars()" procedure in the document. In this example two user-defined toolbars are created:

   ```vba
   Sub CreateApplicationToolbars()
   'VBA6
   'Declare toolbar-objects...:
   Dim objToolbar1 As HMIToolbar
   Dim objToolbar2 As HMIToolbar
   
   'Add the toolbars with parameter "Key"
   Set objToolbar1 = Application.CustomToolbars.Add("AppToolbar1")
   Set objToolbar2 = Application.CustomToolbars.Add("AppToolbar2")
   End Sub
   ```

4. Start the procedure with <F5>.

Result

The two toolbars are inserted beneath the Graphics Designer toolbars.

See also

Add Method (CustomToolbars Listing) (Page 1788)
How to assign VBA macros to menus and toolbars (Page 1642)
How to assign help texts to menus and toolbars (Page 1639)
How to Add a New Icon to the Toolbar (Page 1637)
Configuring Menus and Toolbars (Page 1626)
Creating Customized Menus and Toolbars (Page 1627)
How to Add a New Icon to the Toolbar

Requirement

You must have created the user-defined toolbar first.

Introduction

You can insert two different types of objects in the user-defined toolbar:

- Symbol: To call VBA macros.
- Separator line: For clearer design of user-defined toolbars.

The "Position" parameter determines the order of the icons within the user-defined toolbar.

The "Key" parameter is a unique identification of the icon. This parameter is used if you use the "ToolbarItemClicked" event for calling VBA macros.

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:
3. To create icons in a previously created user-defined toolbar, you can for example insert an "InsertToolbarItems()" procedure in the document. In this example, two icons separated by a separator line are created in the user-defined toolbar "AppToolbar1":

```vba
Sub InsertToolbarItems()
'VBA7
Dim objToolbar1 As HMIToolbar
Dim objToolbarItem1 As HMIToolbarItem

'Add a new toolbar:
Set objToolbar1 = Application.CustomToolbars.Add("AppToolbar1")
'Adds two toolbar-items to the toolbar
'("InsertToolbarItem(Position, Key, DefaultToolTipText)"-Methode):
Set objToolbarItem1 = objToolbar1.ToolbarItems.InsertToolbarItem(1, "tItem1_1", "First Symbol-Icon")
Set objToolbarItem1 = objToolbar1.ToolbarItems.InsertToolbarItem(3, "tItem1_2", "Second Symbol-Icon")

'Adds a separator between the two toolbar-items
'("InsertSeparator(Position, Key)"-Methode):
Set objToolbarItem1 = objToolbar1.ToolbarItems.InsertSeparator(2, "tSeparator1_3")
End Sub
```

4. Start the procedure with <F5>.

**Result**

The "InsertToolbarItems()" procedure adds a toolbar with two icons, separated by a dividing line, to the Graphics Designer toolbars:

![Graphics Designer - tablecontrol.PDL](image)

**Note**

Use the icon property in order to specify a graphic (*.ICO format) for a tool icon.
See also

Creating Customized Menus and Toolbars (Page 1627)
Icon Property (Page 2275)
InsertSeparator Method (Page 1853)
InsertToolbarItem Method (Page 1856)
How to assign VBA macros to menus and toolbars (Page 1642)
How to assign help texts to menus and toolbars (Page 1639)
How to Create an Application-specific Toolbar (Page 1635)
Configuring Menus and Toolbars (Page 1626)

How to assign help texts to menus and toolbars

Requirements

You must have created the user-defined menu or the user-defined toolbar first.

Introduction

When the configuring engineer moves the mouse over a user-defined menu item or over a user-defined icon, you can provide additional help text to explain the functionality in more detail:

- You can define a help text for user-defined menu items and icons; the help text is displayed in the status bar.
- For user-defined icons, the default option is to create the help text as a tooltip.

You can also define status texts and tooltip texts for other languages.
Procedure

1. Open the VBA editor in Graphics Designer (&lt;ALT+F11&gt; or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:

3. To assign a status text to a user-defined menu item, you can for example insert an "AddStatusTextsToAppMenu1()" procedure in the document. In this example one status text in German and one in English is assigned to the first menu item in the previously created "AppMenu1" menu:

   Sub AddStatusTextsToAppMenu1()
      'VBA8
      Dim objMenu1 As HIMenu
      
      'Object "objStatusTextMenuItems1" contains foreign-language texts
      Dim objStatusTextMenuItems1 As HIMLanguageText
      Set objMenu1 = Application.CustomMenus("AppMenu1")
      
      'Assign a statustext to a menuitem:
      objMenu1.MenuItems("mItem1_1").StatusText = "Statustext the first menuitem"
      
      'Assign a foreign statustext to a menuitem:
      Set objStatusTextMenuItems1 = objMenu1.MenuItems("mItem1_1").LDStatusTexts.Add(1033, "This is my first status text in English")
   End Sub
4. To assign status and foreign-language tool tip text to a user-defined icon on the toolbar, insert a "AddStatusAndTooltipTextsToAppToolbar1()" procedure in the document, for example. In this example, the first icon on the toolbar created is assigned a status text (German/English) and an English tool tip text:

```vba
Sub AddStatusAndTooltipTextsToAppToolbar1()
    'VBA9
    Dim objToolbar1 As HMIToolbar
    'Variable "StatusTextToolbarItem1" for foreign statustexts
    Dim objStatusTextToolbarItem1 As HMLanguageText
    'Variable "TooltipTextToolbarItem1 for foreign tooltiptexts
    Dim objTooltipTextToolbarItem1 As HMLanguageText
    Set objToolbar1 = Application.CustomToolbars("AppToolbar1")
    'Assign a statustext to a toolbaritem:
    objToolbar1.ToolbarItems("tItem1_1").StatusText = "Statustext für das erste Symbol-Icon"
    'Assign a foreign statustext to a toolbaritem:
    Set objStatusTextToolbarItem1 = objToolbar1.ToolbarItems("tItem1_1").LDStatusTexts.Add(1033, "This is my first status text in English")
    'Assign a foreign tooltiptext to a toolbaritem:
    Set objTooltipTextToolbarItem1 = objToolbar1.ToolbarItems("tItem1_1").LDTooltipTexts.Add(1033, "This is my first tooltip text in English")
End Sub
```

5. Start the procedure with <F5>.

Results

The status text is displayed when you move the mouse pointer over the user-defined menu item or the icon.

See also

- LDTooltipTexts Property (Page 2303)
- LDStatusTexts Property (Page 2301)
- LanguageTexts Object (Listing) (Page 2000)
- Add Method (Page 1786)
- How to assign VBA macros to menus and toolbars (Page 1642)
- How to Add a New Icon to the Toolbar (Page 1637)
- How to add a new menu entry to a menu (Page 1631)
- Configuring Menus and Toolbars (Page 1626)
- Creating Customized Menus and Toolbars (Page 1627)
How to assign VBA macros to menus and toolbars

Introduction

There are two possible ways for you to assign VBA macros to user-defined menus and toolbars:

- You can use either the VBA event handlers "MenuItemClicked" and "ToolbarItemClicke" or
- "Macro" property.

Note

You will find the VBA code for creating the required user-defined menus and toolbars in this documentation under "Adding a New Menu Item to a Menu" and "Adding a New Icon to a Toolbar".

Procedure

Assigning a VBA macro with a VBA event handler

Note

You will find further information on VBA event handlers in this documentation under "Event Handling".

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:

3. To start a VBA macro via the VBA event handlers, use the "MenuItemClicked" or "ToolbarItemClicke" event:
4. Insert the VBA code from the "VBA10" table.
5. Start the procedure with <F5>. 
Assigning a VBA Macro using the "Macro" property

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:

3. To start a VBA macro via the Macro property, assign the VBA macro to each menu item or icon. In the following example, a user-defined menu with two menu entries is created, which retrieve two different VBA macros:

   The VBA code of the following VBA11 example depends on the file type. The VBA code is added as an example for a PDL and a PDT file. Both cases can be distinguished in the following manner:

   - PDL file:
     The VBA code in a PDL file is only executed when this PDL file is being displayed.

   - PDT file:
     The VBA code in a PDT file is always executed when the Graphics Designer is open.

4. Insert the VBA code from the "VBA11" table. Sample code for PDL file or "VBA821: Sample code for PDT file".

   You can call the following two procedures via the menu items in the user-defined menu "DocMenu1":

5. Insert the VBA code from the "VBA12" table.

6. Start the procedure with <F5>.

The following tables show the VBA codes for the example:
Start VBA via event handler (VBA10)

```vbnet
Option Explicit
'VBA10
'The next declaration has to be placed in the module section
Dim WithEvents theApp As grafexe.Application

Private Sub SetApplication()
  'This procedure has to be executed (with "F5") first
  Set theApp = grafexe.Application
End Sub

Private Sub theApp_MenuItemClicked(ByVal MenuItem As IHMIMenuItem)
  Dim objClicked As HMIMenuItem
  Dim varMenuItemKey As Variant
  Set objClicked = MenuItem
  ' "varMenuItemKey" contains the value of parameter "Key"
  ' from clicked menu-item
  varMenuItemKey = objClicked.Key
  Select Case varMenuItemKey
  Case "mItem1_1"
    MsgBox "The first menuitem was clicked!"
  End Select
End Sub

Private Sub theApp_ToolbarItemClicked(ByVal ToolbarItem As IHMIToolbarItem)
  Dim objClicked As HMIToolbarItem
  Dim varToolbarItemKey As Variant
  Set objClicked = ToolbarItem
  ' "varToolbarItemKey" contains the value of parameter "Key"
  ' from clicked toolbar-item
  varToolbarItemKey = objClicked.Key
  Select Case varToolbarItemKey
  Case "tItem1_1"
    MsgBox "The first symbol-icon was clicked!"
  End Select
End Sub
```
Creating a menu (VBA11: Sample code for PDL file)

Sub CreateDocumentMenusUsingMacroProperty()
'VBA11
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "First Menuitem")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "Second Menuitem")
'Assign a VBA-macro to every menu item
With ActiveDocument.CustomMenus("DocMenu1")
  .MenuItems("dmItem1_1").Macro = "TestMacro1"
  .MenuItems("dmItem1_2").Macro = "TestMacro2"
End With
End Sub

Creating a menu (VBA821: Sample code for PDT file)

Sub CreateDocumentMenusUsingMacroProperty()
'VBA821
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Set objDocMenu = Application.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "First Menuitem")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "Second Menuitem")
'Assign a VBA-macro to every menu item
With Application.CustomMenus("DocMenu1")
  .MenuItems("dmItem1_1").Macro = "TestMacro1"
  .MenuItems("dmItem1_2").Macro = "TestMacro2"
End With
End Sub

Macros for user-defined menu entries (VBA12)

Sub TestMacro1()
'VBA12
MsgBox "TestMacro1 was executed"
End Sub

Sub TestMacro2()
MsgBox "TestMacro2 was executed"
End Sub
See also

Macro Property (Page 2320)
ToolbarItemClicked Event (Page 1782)
MenuItemClicked Event (Page 1775)
How to Add a New Icon to the Toolbar (Page 1637)
How to add a new menu entry to a menu (Page 1631)
Event Handling (Page 1709)
Executing VBA Macros in Graphics Designer (Page 1619)

3.3.2.4 Accessing the component library with VBA

Accessing the component library with VBA

Introduction

The component library contains a large selection of ready-made objects which you can use to design your screens efficiently. The component library consists of a global library and a project-related library:

- The "Global Library" contains prepared objects that are supplied with WinCC. The objects are filed in folders, sorted according to subjects, such as valves, motors, cables and many others.

- The "Project Library" contains neither objects nor folders when you have created a new project. You can create objects which you need only in this particular project in the "Project Library".

VBA gives you full access to the component library: You can create and delete folders and save objects in the component library or insert them into pictures.

Access to the component library with VBA

![Diagram of component library structure]
The component library is represented in VBA by the "SymbolLibraries" listing. The listing contains two elements, which represent the "Global Library" and the "Project Library". The "FolderItems" listing contains elements, which represent folders as well as objects.

**Note**

To address an object in the "SymbolLibraries" listing you use either the index number or the internal name.

You can find out the internal name by clicking the right mouse button on the relevant object in the component library and then choosing the "Copy path" command in the pop-up menu.

The path to the object within the component library is then copied to the clipboard.

**Global Library (1)**

The "Global Library" is the first element in the SymbolLibraries listing, which you address using index number "1". You address the "Project Library" using index number "2".

Access to the "Global Library" with VBA:

```vba
'VBA17
Application.SymbolLibraries(1)
```

**Folder (2)**

A folder in the component library contains either other folders or the objects of a particular subject area. In VBA a folder corresponds to the "FolderItem" object and its type is "Folder". The folders are contained in the "FolderItems" listing. With VBA you can create a new folder or delete an existing one, and add an object to the folder via the clipboard.

Access to the "Plant Components" folder with VBA:

```vba
'VBA18
Application.SymbolLibraries(1).FolderItems("Folder2")
```
Object (3)

In VBA an object corresponds to the "FolderItem" object and its type is "Item". The objects are contained in the "Folder" listing. With VBA you can delete an object or copy it to the clipboard.

Access to the "PC" object with VBA:

```vba
Application.SymbolLibraries(1).FolderItems("Folder2").Folder("Folder2").Folder.Item("Object1").DisplayName
```

Creating or deleting folders in the component library

Use the following methods to create or delete folders:

- "AddFolder(DefaultValue)" Method: Creates a new folder in the components library. A newly created folder receives the internal name "FolderX", where "X" stands for a consecutive number.
- "Delete()" Method: Deletes an existing folder (including all folders and objects that it contains) from the component library.

Inserting or deleting an object in the component library

You can copy objects within the component library (for example from the "Global Library" to the "Project Library"), insert an object from a picture into the component library or delete an object from the component library:

- Methoden "CopyToClipboard()" und "AddFromClipboard()": Copies an object to the clipboard within the component library.
- "AddItem(DefaultValue, pHMIObject)" method: Copies an existing object in the picture into a folder in the component library.
- "Delete()" Method: Deletes an object.

Finding an object or folder in the component library

Use the "FindByDisplayName("DisplayName") method to search for an object or folder. The specified display name is dependent on which language is currently set. The search ends with the first occurrence of the object or folder that you are looking for.

Inserting an object into a picture from the component library

Use the "CopyToClipboard()" and "PasteClipboard()" methods to insert an object from the component library into the current picture.

See also

CopyToClipboard Method (Page 1824)
PasteClipboard Method (Page 1868)
Introduction

Here you will find the following instructions for editing the component library with VBA:

- Creating a new folder
- Copying an object within the component library
- Copying an object from the active picture into the component library
- Deleting an object from the component library

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:
3. To create a new folder in the component library, you can for example insert an "AddNewFolderToProjectLibrary()" procedure in the document. In this example the folder "My folder" is created:

```vba
Sub AddNewFolderToProjectLibrary()
    'VBA20
    Dim objProjectLib As HMISymbolLibrary
    Set objProjectLib = Application.SymbolLibraries(2)

    '("AddFolder(DefaultValue)"-Methode):
    objProjectLib.FolderItems.AddFolder("Custom Folder")
End Sub
```

4. In order to copy an object from the "global library" to the "library project", insert a "CopyObjectFromGlobalLibraryToProjectLibrary()" procedure in the document, for example. In this example, the object "Object1" is copied:

```vba
Sub CopyObjectFromGlobalLibraryToProjectLibrary()
    'VBA21
    Dim objGlobalLib As HMISymbolLibrary
    Dim objProjectLib As HMISymbolLibrary
    Set objGlobalLib = Application.SymbolLibraries(1)
    Set objProjectLib = Application.SymbolLibraries(2)

    'Copies object "PC" from the "Global Library" into the clipboard
    objGlobalLib.FolderItems("Folder2").Folder("Folder2").Folder.Item("Object1").CopyToClipboard

    'The folder "Custom Folder" has to be available
    objProjectLib.FolderItems("Folder1").Folder.AddFromClipBoard("Copy of PC/PLC")
End Sub
```

5. In order to copy an object from the active picture to the "Project Library", insert a procedure like "AddObjectFromPictureToProjectLibrary()" into document. In this example, the object "Circle1" is created in the active picture and then copied to the folder "Folder1":

```vba
Sub AddObjectFromPictureToProjectLibrary()
    'VBA22
    Dim objProjectLib As HMISymbolLibrary
    Dim objCircle As HMICircle
    Set objProjectLib = Application.SymbolLibraries(2)

    'Insert new object "Circle1"
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")

    'The folder "Custom Folder" has to be available
    '("AddItem(DefaultValue, pHMIObject)"-Methode):
    objProjectLib.FolderItems("Folder1").Folder.AddItem "ProjectLib Circle", ActiveDocument.HMIObjects("Circle1")
End Sub
```
6. To delete an object from the component library, insert a "DeleteObjectFromProjectLibrary()" procedure in the document, for example. In this example the previously created folder "Folder1" is deleted:

```
Sub DeleteObjectFromProjectLibrary()
    'VBA23
    Dim objProjectLib As HMISymbolLibrary
    Set objProjectLib = Application.SymbolLibraries(2)

    'The folder "Custom Folder" has to be available
    "Delete" Method:
    objProjectLib.FolderItems("Folder1").Delete
End Sub
```

7. Start the procedure with <F5>.

See also

- SymbolLibrary Object (Page 2074)
- SymbolLibraries Object (Listing) (Page 2073)
- PasteClipboard Method (Page 1868)
- Delete Method (Page 1831)
- CopyToClipboard Method (Page 1824)
- AddItem Method (Page 1802)
- AddFromClipboard Method (Page 1799)
- AddFolder Method (Page 1798)
- How to paste an object from the object library into a picture with VBA (Page 1652)
- Accessing the component library with VBA (Page 1646)
How to paste an object from the object library into a picture with VBA

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:
3. To insert an object from the "Global Library" into the active picture, you can for example insert a "CopyObjectFromGlobalLibraryToActiveDocument()" procedure in the document. In this example the object "Object1" is inserted

```vba
Sub CopyObjectFromGlobalLibraryToActiveDocument()
'VBA24
Dim objGlobalLib As HMISymbolLibrary
Dim objHMIObject As HMIObject
Dim iLastObject As Integer
Set objGlobalLib = Application.SymbolLibraries(1)

'Copy object "PC" from "Global Library" to clipboard
objGlobalLib.FolderItems("Folder2").Folder("Folder2").Folder.Item("Object1").CopyToClipboard

'Get object from clipboard and add it to active document
ActiveDocument.PasteClipboard

'Get last inserted object
iLastObject = ActiveDocument.HMIObjects.Count
Set objHMIObject = ActiveDocument.HMIObjects(iLastObject)

'Set position of the object:
With objHMIObject
 .Left = 40
 .Top = 40
End With
End Sub
```

4. Start the procedure with <F5>.

See also

- PasteClipboard Method (Page 1868)
- CopyToClipboard Method (Page 1824)
- How to edit the component library with VBA (Page 1649)
- Accessing the component library with VBA (Page 1646)
3.3.3 Editing Pictures with VBA

3.3.3.1 Editing Pictures with VBA

Introduction

Pictures visualize the process to be executed and observed. They display the important process steps or plant parts and present the production process in a schematic manner. In VBA the picture is represented by the Document object.

![Diagram of Document, Layer, View, Menu, Toolbar hierarchy]

Picture-specific menus and toolbars

In contrast with the application-specific menus and toolbars, the picture-specific menus and toolbars are coupled to a specific picture. The picture-specific menus and toolbars remain visible for as long as the picture is active.

You should use picture-specific menus and toolbars when the called VBA macros are only used in that picture.

Layers

You can access the layers in the Graphics Designer with VBA. Each layer is represented by the Layer object. By changing the properties of the Layer object you can specify among other things the layer names and the zoom settings.

You control the visibility of the RT layers via the Document object. You control the visibility of the CS layers via the View object.

Copies of the picture

You can create copies of a picture with VBA in order to display different views of a picture. The copy of a picture is represented in VBA by the View object.
In the properties of the View object you can among other things set the zoom factor and specify which picture section is to be displayed.

**Note**

If you want to run VBA code in a picture saved in WinCC V7.0 SP1 under WinCC V7.0, you need to deactivate the "CCHMIDotNetObj 1.0 Type Library" in the VBA Editor under "Tools > References".

The VBA program will then be executed within the usual functional scope of WinCC V7.0. In this case, you cannot use the new functions of WinCC V7.0 SP1.

---

**See also**

- Editing a Copy of a Picture with VBA (Page 1658)
- How to Create Picture-specific Menus and Toolbars (Page 1655)
- Editing Layers with VBA (Page 1657)
- Editing Objects with VBA (Page 1660)
- Adapting the Graphics Designer with VBA (Page 1623)

---

### 3.3.3.2 How to Create Picture-specific Menus and Toolbars

**Introduction**

Picture-specific menus and toolbars are linked to a specific picture and remain visible for as long as the picture is open. You should use picture-specific menus and toolbars when the VBA macros used there are relevant only for that particular picture.

**Procedure**

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor").
2. Open the document "ThisDocument" in the Project Explorer:
3. To create a picture-specific menu, you can for example insert a "CreateDocumentMenus()" procedure in the document "ThisDocument":

```vba
Sub CreateDocumentMenus()
'VBA25
'declare menuobjects:
Dim objMenu1 As HMIMenu
Dim objMenu2 As HMIMenu
'insert menus ("InsertMenu"-methode) with
'parameters - "Position", "Key", "DefaultLabel":
Set objMenu1 = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
Set objMenu2 = ActiveDocument.CustomMenus.InsertMenu(2, "DocMenu2", "Doc_Menu_2")
End Sub
```

4. In order to create a picture-specific toolbar, insert a procedure like "CreateDocumentToolbars()" into the document "ThisDocument":

```vba
Sub CreateDocumentToolbars()
'VBA26
'declare required number of toolbarobjects:
Dim objToolbar1 As HMIToolbar
Dim objToolbar2 As HMIToolbar
'insert toolbars ("Add"-methode) with
'parameter - "Key":
Set objToolbar1 = ActiveDocument.CustomToolbars.Add("DocToolbar1")
Set objToolbar2 = ActiveDocument.CustomToolbars.Add("DocToolbar2")
End Sub
```

5. Always start the procedure with <F5>.

See also

- Add Method (CustomToolbars Listing) (Page 1788)
- InsertMenu Method (Page 1850)
- How to assign VBA macros to menus and toolbars (Page 1642)
- How to assign help texts to menus and toolbars (Page 1639)
- How to Add a New Icon to the Toolbar (Page 1637)
- How to create menus in multiple languages (Page 1633)
- How to add a new menu entry to a menu (Page 1631)
- Creating Customized Menus and Toolbars (Page 1627)
- Configuring Menus and Toolbars (Page 1626)
3.3.3.3 Editing Layers with VBA

Introduction

You can arrange objects in 32 layers in the Graphics Designer. The layers are differentiated according to CS layers and RT layers so that the visibility of the layers in the picture (CS) and in runtime (RT) can be controlled separately. In VBA a layer is represented by the Layer object:

![Layer Object Diagram]

In the Graphics Designer the lowest layer is "Layer 0". To give back the lowest layer with VBA, use the index "1":

```vba
ActiveDocument.Layers(1)
```

Using the Layer object

You use the Layer object in order to specify the minimum and maximum zoom for a layer and to assign a name. In the following example the settings for the lowest layer are configured in the active picture:

```vba
Sub ConfigureSettingsOfLayer
    'VBA27
    Dim objLayer As HMILayer
    Set objLayer = ActiveDocument.Layers(1)
    With objLayer
        'Configure "Layer 0"
        .MinZoom = 10
        .MaxZoom = 100
        .Name = "Configured with VBA"
    End With
End Sub
```

Controlling the visibility of CS and RT layers

You control the visibility of the CS layers via the View object. Use the Document object in order to determine which layers are to be displayed or hidden in runtime. You can control the visibility of the CS and RT layers with the following methods:

- Methode "IsCSLayerVisible(Index)"; Checks whether the specified CS layer is displayed.
- Methode "SetCSLayerVisible(Index, Val)"; Shows or hides the specified CS layer.

Use the IsRTLayerVisible and SetRTLayerVisible methods for the RT layers in the same way.
3.3.3.4 Editing a Copy of a Picture with VBA

Introduction

You can create copies of a picture with VBA in order to display different views of a picture. Each view is shown in a separate window. The copy of a picture is represented in VBA by the View object:

```
Document
    Views
        View
```

In the properties of the View object you can among other things set the zoom factor and specify which picture section is to be displayed.

Creating a copy of a picture

Use the Add method to create a copy of the specified picture. In this example a copy of the active picture is created and activated:

```vba
Sub CreateAndActivateView()
    'VBA28
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
End Sub
```
Editing a copy of a picture

You can edit each copy of a screen as follows:

- Adjust zoom factor: Use the zoom property.
- Specify picture zoom area: Specify the picture section: use the "ScrollPosX" and "ScrollPosY" properties to specify the picture zoom area using the scroll bars.
- Showing and Hiding CS layers: You can use the SetCSLayerVisible(Index) method for example to show or hide the specified layer. You can select the layer on which you want to edit the objects with the ActiveLayer property.

In the following example a copy of the active picture is created and activated. The zoom factor is set to 150% and the position of the scrollbars is changed:

```vba
Sub SetZoomAndScrollPositionInActiveView()
    'VBA29
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    With objView
        .Activate
        .ScrollPosX = 40
        .ScrollPosY = 10
        .Zoom = 150
    End With
End Sub
```

See also

- Add Method (Views Listing) (Page 1793)
- ScrollPosY Property (Page 2411)
- ScrollPosX Property (Page 2410)
- ActiveLayer Property (Page 2108)
- View Object (Page 2102)
- SetCSLayerVisible Method (Page 1885)
- IsCSLayerVisible Method (Page 1857)
- Activate Method (Page 1785)
- Editing Layers with VBA (Page 1657)
- Editing Pictures with VBA (Page 1654)
3.3.4 Editing Objects with VBA

3.3.4.1 Editing Objects with VBA

Access to objects in the Graphics Designer

In VBA all object types of the current picture are contained in the "HMIObjects" listing. They are not divided according to object type (Standard, Smart, Windows and Controls objects) as in the Graphics Designer. With VBA you can therefore run through all objects in one or more pictures with a loop.

When you have selected objects in the picture, these objects are contained in the "Selection" listing. Use the "HMIDefaultObjects" listing if you want to change the default settings of the properties of an object.

To address an object in a picture with VBA, use either the object name, e.g. "ActiveDocument.HMIObjects("Circle1"), or the index number. "ActiveDocument.HMIObjects(1)" references for example the first object in the active picture.

Editing objects with VBA

You have the following possibilities for editing objects with VBA:

- Create a new object in a picture
- Delete an existing object
- Copy an existing object
- Group existing objects or cancel the grouping
- Search for objects
- Display or change object properties

When you insert a new object into a picture with VBA, the object behaves in the same way as if you double-clicked it in the Graphics Designer object palette.

The object is given the predefined property values and is inserted in the top left-hand corner of the picture.
Access to the object properties is dependent on how you created the object. Two examples illustrate this:

**Example 1:**

In this example a circle of the type "HMIObject" is inserted into the current picture. You can use a VBA object of the "HMIObject" type or all objects in the Graphics Designer. However, you have to address individual properties of the respective object explicitly via the "Properties(Index)" property:

```
Sub AddObject()
  'VBA30
  Dim objObject As HMIObject
  Set objObject = ActiveDocument.HMIObjects.AddHMIObject("CircleAsHMIObject", "HMICircle")
  'standard-properties (e.g. the position) are available every time:
  objObject.Top = 40
  objObject.Left = 40
  'non-standard properties can be accessed using the Properties-collection:
  objObject.Properties("FlashBackColor") = True
End Sub
```

**Example 2:**

In this example a circle of the type "HMICircle" is inserted into the current picture. In contrast with Example 1 you can only use the "objCircle" object for objects of the "HMICircle" type, however:

```
Sub AddCircle()
  'VBA31
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("CircleAsHMICircle", "HMICircle")
  'The same as in example 1, but here you can set/get direct the specific properties of the circle:
  objCircle.Top = 80
  objCircle.Left = 80
  objCircle.FlashBackColor = True
End Sub
```

See also

- LanguageFonts Object (Listing) (Page 1997)
- VBA Reference (Page 1745)
- Underlined Property (Page 2460)
- Size Property (Page 2419)
- Parent Property (Page 2359)
3.3.4.2 Default objects, Smart objects, Windows objects and Tube objects

Default objects, Smart objects, Windows objects and Tube objects

Introduction

You use the Standard, Smart and Windows objects to design your pictures. In the Graphics Designer, these objects are available in the “Default” selection window:

![Image of object selection windows]

VBA enables you to access these objects in all pictures in your project. If, for example, you want to change the background color of all circles in a project with several pictures, you can do this with a VBA macro.
Paste Object into Picture

Use the "AddHMIObject(ObjectName, ProgID)" method to insert a new object in a picture. ObjectName stands for the name of the object (e.g. "my Circle"), and ProgID for the VBA object designation (e.g. "HMICircle"): 

Sub AddCircle()
'VBA32
'Creates object of type "HMICircle"
Dim objCircle As HMICircle
'
'Add object in active document
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("My Circle", "HMICircle")
End Sub

Edit Object

VBA gives you access to all object properties, which you can edit via the object's properties dialog. You can change and output object properties, and select objects in the picture. If you have not selected an object, you can use the following methods:

- "Find()" method: Searches for an object in the "HMIObjects" listing
- "Delete()" Method: Deletes an HMIObject object.

If you have selected objects, you can edit them via the "Selection" listing with the following methods, among others:

- "AlignLeft()", "AlignRight()", "AlignTop()", "AlignBottom()": These methods align objects.
- "CreateGroup()", "CreateCustomizedObject()": These methods create a group object or customized object.
- "DeselectAll()" method: Cancels the selection of all objects

Remove VBA: references with "Nothing"

Always remove the references used for the Controls, the standard objects, and for the document after you closed the document. For this purpose, set the objects to "Nothing". The following example shows the code for a Control:

Public Sub DrawNewControl
Const strFct = "CreatePdls"
Dim objControl As HMIObject
Dim objDoc As Document
On Local Error GoTo errorhandler
'open the document
'create new object
Set objControl = objDoc.HMIObjects.AddActiveXControl("Control1", "CCAxUserArchiveControl.AxUserArchiveControl.1")
If objControl Is Nothing Then
    GoTo errorhandler
End Sub
End If
' doing something with the control
''
' delete reference to new control
Set objControl = Nothing
'saving PDL and deleting reference to it
objDoc.Save
objDoc.Close
Set objDoc = Nothing
Exit Sub
' errorhandler
errorhandler:
If MsgBox("Error occurred" & vbCrLf & "Yes - resume next" & vbCrLf & "No - stop script", vbOKCancel + vbCritical, strFct) = vbOK Then
    Resume Next
End If
End Sub

See also

Parent Property (Page 2359)
Item Property (Page 2283)
Count Property (Page 2201)
Application Property (Page 2121)

How to edit Default objects, Smart objects, Windows objects and Tube objects

Introduction

Here you will find the following instructions for editing Standard, Smart and Windows objects:

- Define properties of a specific object
- Define properties of a nonspecific object
- Select an object in the active picture
- Find objects in the active picture
- Delete object
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. Open the document "ThisDocument" in the Project Explorer:

3. To define the properties of a specific object type (e.g. "HMICircle"), you can for example insert an "EditDefinedObjectType()" procedure in the document "ThisDocument". In this example a circle is inserted in the active picture and its line weight and color are modified:

```vba
Sub EditDefinedObjectType()
'VBA33
Dim objCircle As HMICircle
Set objCircle =
ActiveDocument.HMIObjects.AddHMIObject("myCircleAsCircle", 
"HMICircle")
With objCircle
'direct calling of object properties available
.BorderStyle = 4
.BorderColor = RGB(255, 0, 255)
End With
End Sub
```

4. To change the properties of a nonspecific object type ("HMIObject"), insert a "EditHMIObject()" procedure in the document "ThisDocument", for example. In this example a circle is inserted in the active picture and its line weight and color are modified:

```vba
Sub EditHMIObject()
'VBA34
Dim objObject As HMIObject
Set objObject =
ActiveDocument.HMIObjects.AddHMIObject("myCircleAsObject", 
"HMICircle")
With objObject
'Access to object properties only with property "Properties":
.Properties("BorderStyle") = 4
.Properties("BorderColor") = RGB(255, 0, 0)
End With
End Sub
```
5. To select an object in the current picture, insert a "SelectObject()" procedure in the document "ThisDocument", for example. In this example, a circle will be inserted in the active picture and selected:

```vba
Sub SelectObject()
'VBA35
Dim objObject As HMIObject
Set objObject = ActiveDocument.HMIOObjects.AddHMIObject("mySelectedCircle", "HMICircle")
ActiveDocument.HMIOObjects("mySelectedCircle").Selected = True
End Sub
```
6. To search for an object in the current picture, insert a "FindObjectByName()", "FindObjectByType()", or "FindObjectByProperty()" procedure in the document "ThisDocument", for example. In this example, objects containing the string "Circle" in their name are searched for:

Sub FindObjectsByName()
    'VBA36
    Dim colSearchResults As HMIICollection
    Dim objMember As HMIObject
    Dim iResult As Integer
    Dim strName As String

    'Wildcards (?, *) are allowed
    Set colSearchResults = ActiveDocument.HMIObjects.Find(ObjectName:="*Circle*")
    For Each objMember In colSearchResults
        iResult = colSearchResults.Count
        strName = objMember.ObjectName
        MsgBox "Found: " & CStr(iResult) & vbCrLf & "Objectname: " & strName
    Next objMember
End Sub

In this example a search is run in the active picture for objects of the type "HMICircle":

Sub FindObjectsByType()
    'VBA37
    Dim colSearchResults As HMIICollection
    Dim objMember As HMIObject
    Dim iResult As Integer
    Dim strName As String

    Set colSearchResults = ActiveDocument.HMIObjects.Find(ObjectType:="HMICircle")
    For Each objMember In colSearchResults
        iResult = colSearchResults.Count
        strName = objMember.ObjectName
        MsgBox "Found: " & CStr(iResult) & vbCrLf & "Objektname: " & strName
    Next objMember
End Sub

In this example a search is run in the active picture for objects with the property "BackColor":

Sub FindObjectsByProperty()
    'VBA38
    Dim colSearchResults As HMIICollection
    Dim objMember As HMIObject
    Dim iResult As Integer
    Dim strName As String

    Set colSearchResults = ActiveDocument.HMIObjects.Find(PropertyName:="BackColor")
    For Each objMember In colSearchResults
        iResult = colSearchResults.Count
        strName = objMember.ObjectName
        MsgBox "Found: " & CStr(iResult) & vbCrLf & "Objectname: " & strName
    Next objMember
End Sub
7. To delete an object, you can for example insert a "DeleteObject()" procedure in the document "ThisDocument". In this example the first object in the active picture will be deleted.

```vba
Sub DeleteObject()
    'VBA39
    'Delete first object in active document:
    ActiveDocument.HMIObjects(1).Delete
End Sub
```

8. Start the procedure with <F5>.

See also

- Find Method (Page 1841)
- Delete Method (Page 1831)
- AddHMIObject Method (Page 1801)
- How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1664)
- Editing Objects with VBA (Page 1660)
OLE Objects

Introduction

You can use VBA to insert OLE Elements into a picture. The OLE Element belongs to the Smart objects. In the Graphics Designer, the object is available in the "Default" selection window:
Paste OLE Element in Picture

Use the "AddOLEControl(ObjectName, ServerName, [CreationType], [UseSymbol]) method to insert an OLE Element into a picture. ObjectName" stands for the object name, and "ServerName" for the application that is to be contained in the OLE Element. The "ServerName" parameter corresponds to the object type in the "object insertion dialog. The last two parameters are optional and represent the possible settings" in the dialog displayed:

![OLE control dialog](image)

You will find further information on the parameters in this documentation under "AddOLEObject method".

In the following example an OLE Element containing a Wordpad document will be inserted into the active picture:

```vba
Sub AddOLEObjectToActiveDocument()
  'VBA40
  Dim objOLEObject As HMIOLEObject
  Set objOLEObject = ActiveDocument.HMIObjects.AddOLEObject("MS Wordpad Document1", "Wordpad.Document.1")
End Sub
```

The OLEObject object is added to the "HMIObjects" listing as the last element and inherits the properties of the HMIOObject object.

See also

- OLEObject Object (Page 2023)
- AddOLEObject Method (Page 1803)
- How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1664)
- Default objects, Smart objects, Windows objects and Tube objects (Page 1662)
- Editing Objects with VBA (Page 1660)
ActiveX controls

Introduction

You can use VBA to insert ActiveX controls into a picture. In the Graphics Designer, you can find the ActiveX controls supplied with WinCC in the "Controls" selection window:

Further information is provided under "AddActiveXControl method" in this documentation and under "Working with controls" in the WinCC documentation.

Integrating standard ActiveX controls

As well as the ActiveX controls supplied with WinCC, you can insert all standard ActiveX controls registered in the operating system into a picture. This means that you also have the option of using ActiveX controls that you have programmed yourself in your pictures. A list of the standard ActiveX controls tested with WinCC is given in the WinCC documentation.
Inserting an ActiveX control into a picture

Use the "AddActiveXControl(ObjectName, ProgID)" method to insert a new ActiveX control into a picture. ObjectName stands for the name of the ActiveX control (e.g. "WinCC Gauge"), and "ProgID" for the VBA object designation (e.g. "XGauge.XGauge.1"):

```vba
Sub AddActiveXControl()
'VBA41
Dim objActiveXControl As HMIActiveXControl
Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge", "XGAUGE.XGaugeCtrl.1")
End Sub
```

The ActiveXControl object is added to the "HMIObjects" listing as the last element and inherits the properties of the HMIObject object.

Access to the properties of the ActiveX control

You must address the object-specific properties of the ActiveX control via the "Properties(Index)" property. You can find out which properties a ActiveX control possesses from the "Object Properties" dialog in the Graphics Designer or from the Properties listing:

```vba
Sub AddActiveXControl()
'VBA42
Dim objActiveXControl As HMIActiveXControl
Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge2", "XGAUGE.XGaugeCtrl.1")

' move ActiveX-control:
objActiveXControl.Top = 40
objActiveXControl.Left = 60

' Change individual property:
objActiveXControl.Properties("BackColor").value = RGB(255, 0, 0)
End Sub
```

Restricted access to background graphics of Controls

The background graphic cannot be configured in VBA for the following Controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCC Digital/Analog Clock Control</td>
<td>Background graphic</td>
</tr>
<tr>
<td>WinCC Gauge Control</td>
<td>Background picture</td>
</tr>
<tr>
<td></td>
<td>Frame picture</td>
</tr>
<tr>
<td>WinCC Push Button Control</td>
<td>PictureSelected</td>
</tr>
<tr>
<td></td>
<td>PictureUnselected</td>
</tr>
<tr>
<td>WinCC slider control</td>
<td>Background picture</td>
</tr>
<tr>
<td></td>
<td>Slider picture</td>
</tr>
</tbody>
</table>
See also

ActiveXControl Object (Page 1900)
AddActiveXControl Method (Page 1795)
How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1664)
Default objects, Smart objects, Windows objects and Tube objects (Page 1662)
Editing Objects with VBA (Page 1660)

.Net controls

Introduction

You can use VBA to insert .Net controls into a picture. In the Graphics Designer, you can find the .Net controls in the "Controls" selection window.

Additional information is provided under "AddDotNetControl method" in this documentation and under "Creating process pictures > Working with controls > .Net controls" in the WinCC documentation.

Inserting a .Net control into a picture

Use the "AddDotNetControl(ObjectName, ControlType, InGAC, AssemblyInfo)" method to insert a new .Net control into a picture. "ObjectName" represents the name of the .Net control. "ControlType" shows the name space of the object. If "InGAC" is "TRUE", the object is registered in the Global Assembly Cache and the associated information is available in "AssemblyInfo".

Sub AddDotNetControl()
"VBA851
Dim DotNetControl As HMIDotNetControl
Set DotNetControl = ActiveDocument.HMIObjects.AddDotNetControl("MyVBAControl",
Culture=neutral, PublicKeyToken=b77a5c561934e089")
End Sub

The .Net control object is added to the "HMIObjects" listing as an element and inherits the properties of the HMIObject object.

Access to the properties of the .Net control

You can find out which properties a .Net control has from the "Object Properties > Control Properties" dialog in Graphics Designer.
WPF controls

Introduction
You can use VBA to insert WPF controls into a picture. In the Graphics Designer, you can find the WPF controls in the "Controls" selection window.

Additional information is provided under "AddWPFControl method" in this documentation and under "Creating process pictures > Working with controls > WPF controls" in the WinCC documentation.

Inserting a WPF control into a picture
Use the "AddWPFControl(ObjectName, ControlType, InGAC, AssemblyInfo)" method to insert a new WPF control into a picture. "ObjectName" represents the name of the .Net control. "ControlType" shows the name space of the object. If "InGAC" is "TRUE", the object is registered in the Global Assembly Cache and the associated information is available in "AssemblyInfo".

Sub AddWPFControl()
'VBA852
Dim WPFControl As HMIWPFControl
Set WPFControl = ActiveDocument.HMIObjects.AddWPFControl("MyWPFVBAControl", "WinCCWPFControl.TestControl", False, "Assembly=Z:\TestControl\WinCCWPFControl.dll")
End Sub

The ActiveXControl object is added to the "HMIObjects" listing as an element and inherits the properties of the HMIObj ect object.

Access to the properties of the WPF control
You can find out which properties a WPF control has from the "Object Properties > Control Properties" dialog in Graphics Designer.

3.3.4.3 Group Objects

Group Objects

Introduction
With VBA you can create a group object from selected objects in the Graphics Designer. You can add objects to the group object, or remove objects, without having to ungroup the group object itself. You have unrestricted access to the object properties of the individual objects in the group object. You can also ungroup a group object again, or delete it entirely.
The following object types cannot be part of a group object:

- CustomizedObject (Customized object)
- ActiveXControl
- OLEObject

Further information regarding group objects can be found in the WinCC documentation under "Group Object".

Creating a group object

To create a group object, select the objects that you want to be part of the group object in the Graphics Designer. The selected objects are then contained in the "Selection" listing. You create the group with the "CreateGroup()" method:

```vba
Sub CreateGroup()
    'VBA43
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim objGroup As HMIGroup
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    Set objGroup = ActiveDocument.Selection.CreateGroup
    objGroup.ObjectName = "myGroup"
End Sub
```

The group object is inserted at the end of the "HMIObjects" listing. The objects that are contained in the group object retain their index numbers and continue to be available in the "HMIObjects" listing.

The objects in the group object are also included in the "GroupedHMIObjects" listing, although the index numbers are reassigned.

Give the group object a name (objGroup.Name = "My Group") so that you can uniquely identify it. If you do not assign a name, the group object is given the default designation for the group object (e.g. "Group1").

The group object has the same properties as the objects of the "Object" type.
Editing a group object

You can edit a group object as follows:

- Methode "Add(Index)" : Adds a new object to the group object.
- Methode "Remove(Index)" : Removes a object from the group object.
- "UnGroup()" method: Ungroups the group object (ungroup).
- "Delete()" Method: Deletes the group object and the objects that it contains.

Editing objects in a group object

Use the "GroupedHMIObjects" listing in order to select an object in the group object. In order to access its object property you must access the name of the object property via the "Properties" property, for example:

```vba
Sub ModifyPropertyOfObjectInGroup()
'VBA44
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("myGroup")
objGroup.GroupedHMIObjects(1).Properties("BorderColor") = RGB(255, 0, 0)
End Sub
```

See also

- SelectedObjects object (Listing) (Page 2060)
- GroupedObjects Object (Listing) (Page 1983)
- Ungroup Method (Page 1892)
- Remove Method (Page 1871)
- Delete Method (Page 1831)
- Add Method (GroupedObjects Listing) (Page 1791)
- How to Edit Objects in Group Objects Using VBA (Page 1679)
- How to Edit the Group Objects Using VBA (Page 1676)
- Editing Objects with VBA (Page 1660)
- VBA in the Graphics Designer (Page 1621)

How to Edit the Group Objects Using VBA

Requirements

You must have created at least two graphic objects in the Graphics Designer and you must have selected them.
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor").

2. Open the document "ThisDocument" in the Project Explorer:

3. To create a group object from selected objects, you can for example insert a "CreateGroup()" procedure in the document "ThisDocument". In this example the group object "My Group" is created from a number of objects.

   ```vba
   Sub CreateGroup()
   'VBA45
   Dim objCircle As HMICircle
   Dim objRectangle As HMIRectangle
   Dim objGroup As HMIGroup
   Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
   Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
   With objCircle
   .Top = 40
   .Left = 40
   .Selected = True
   End With
   With objRectangle
   .Top = 80
   .Left = 80
   .Selected = True
   End With
   MsgBox "Objects selected!"
   Set objGroup = ActiveDocument.Selection.CreateGroup
   'The name identifies the group-object
   objGroup.ObjectName = "My Group"
   End Sub
   ```
4. To add an object to the "My Group" group object, insert a "AddObjectToGroup()" procedure in the document "ThisDocument", for example. In this example, an ellipse is added to the "My Group" group object:

Sub AddObjectToGroup()
'VBA46
Dim objGroup As HMIGroup
Dim objEllipseSegment As HMIEllipseSegment
'Adds new object to active document
Set objEllipseSegment =
ActiveDocument.HMIObjects.AddHMIObject("EllipseSegment", "HMIEllipseSegment")
Set objGroup = ActiveDocument.HMIObjects("My Group")
'Adds the object to the group
objGroup.GroupedHMIObjects.Add ("EllipseSegment")
End Sub

5. To remove an object from the "My Group" group object, insert a "RemoveObjectFromGroup()" procedure in the document "ThisDocument", for example. In this example the first object will be removed from the "My Group" group object:

Sub RemoveObjectFromGroup()
'VBA47
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("My Group")
'delete first object of the group-object
objGroup.GroupedHMIObjects.Remove (1)
End Sub

6. To ungroup the "My Group" group object again, insert a procedure "UnGroup()" into the document "ThisDocument". In this example, the "My Group" group object is ungrouped:

Sub UnGroup()
'VBA48
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("My Group")
objGroup.UnGroup
End Sub

7. To delete the "My Group" group object, insert a procedure "DeleteGroup()" into the document "ThisDocument". In this example, the "My Group" group object is deleted, together with the objects it contains:

Sub DeleteGroup()
'VBA49
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("My Group")
objGroup.Delete
End Sub

8. Always start the procedure with <F5>.

See also

Ungroup Method (Page 1892)
Remove Method (Page 1871)
Delete Method (Page 1831)
How to Edit Objects in Group Objects Using VBA

Introduction

Here you will find the following instructions for editing objects in a group object with VBA:

- Editing a property of an object in the group object
- Editing a property of all objects in the group object

Requirement

You must have created at least two graphic objects in the Graphics Designer and you must have grouped them.

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor").
2. Open the document "ThisDocument" in the Project Explorer:
3. To edit a property of an object within the group object, you can for example insert a "ChangePropertiesOfGroupMembers()" procedure into the document "ThisDocument". In this example the properties of three different objects are modified in the group object "My Group":

```vba
Sub ChangePropertiesOfGroupMembers()
'VBA50
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
With objEllipse
  .Top = 120
  .Left = 120
  .Selected = True
End With
MsgBox "Objects selected!"
Set objGroup = ActiveDocument.Selection.CreateGroup
objGroup.ObjectName = "My Group"
'Set bordercolor of 1. object = "red":
objGroup.GroupedHMIObjects(1).Properties("BorderColor") = RGB(255, 0, 0)
' set x-coordinate of 2. object = "120" :
objGroup.GroupedHMIObjects(2).Properties("Left") = 120
' set x-coordinate of 3. object = "90" :
objGroup.GroupedHMIObjects(3).Properties("Top") = 90
End Sub
```
4. To change the properties of all the objects in the group object, insert a "ChangePropertiesOfAllGroupMembers()" procedure in the document. In this example, the "BorderColor" property of each object in the "My Group" group object is changed. This example will not work unless you have created the "My Group" group object:

```vba
Sub ChangePropertiesOfAllGroupMembers()
'VBA51
Dim objGroup As HMIGroup
Dim iMaxMembers As Integer
Dim iIndex As Integer
Set objGroup = ActiveDocument.HMIObjects("My Group")
iIndex = 1

'Get number of objects in group-object:
iMaxMembers = objGroup.GroupedHMIObjects.Count

' set linecolor of all objects = "yellow":
For iIndex = 1 To iMaxMembers
    objGroup.GroupedHMIObjects(iIndex).Properties("BorderColor") = RGB(255, 255, 0)
Next iIndex
End Sub
```

5. Always start the procedure with <F5>.

See also

- Properties Object (Listing) (Page 2041)
- GroupedObjects Object (Listing) (Page 1983)
- Ungroup Method (Page 1892)
- Remove Method (Page 1871)
- Delete Method (Page 1831)
- Add Method (GroupedObjects Listing) (Page 1791)
- How to Edit the Group Objects Using VBA (Page 1676)
- Group Objects (Page 1674)
- Editing Objects with VBA (Page 1660)
- VBA in the Graphics Designer (Page 1621)
3.3.4.4 Customized Objects

Customized Objects

Introduction

You can use VBA to create a customized object from selected objects in the Graphics Designer. In contrast to the group object, in the case of a customized object only those object properties are available which you have selected in the "Configuration Dialog" for the customized object. It is not possible to configure a customized object with VBA.

Further information regarding customized objects can be found in the WinCC documentation under "Customized Object".

Creating a customized object with VBA

Use the "CreateCustomizedObject()" method to create a customized object from selected objects:

Sub CreateCustomizedObject()
    'VBA52
    Dim objCustomizedObject As HMICustomizedObject
    Set objCustomizedObject = ActiveDocument.Selection.CreateCustomizedObject
    objCustomizedObject.ObjectName = "My Customized Object"
End Sub

When you apply the "CreateCustomizedObject()" method, the "Configuration Dialog" appears in which you select the object properties. The customized object that you have created is added to the "HMIObjects" listing. Give the customized object an appropriate name (objCustomizedObject.Name = "My Customized Object") so that you can uniquely identify it.

Note

If you open a document as invisible, do not create a user object there with a VBA script. Program execution will otherwise be interrupted by a configuration dialog.

Editing Customized Objects

You can edit a customized object as follows:

- "Destroy" method: Ungroups the customized object.
- "Delete" Method: Deletes the customized object and the objects that it contains.
Editing objects in a customized object

Use the "Properties" property to access the selected object properties of the objects contained in the customized object.

Sub EditCustomizedObjectProperty()
    'VBA53
    Dim objCustomizedObject As HMICustomizedObject
    Set objCustomizedObject = ActiveDocument.HMIObjects(1)
    objCustomizedObject.Properties("BackColor") = RGB(255, 0, 0)
End Sub

If you have selected more than one identical property (for example the background color of a circle and of a rectangle), these properties will be numbered ("BackColor" and "BackColor1").

See also

HMIOBJECT Object (Page 1988)
CustomizedObject Object (Page 1938)
Destroy Method (Page 1835)
Delete Method (Page 1831)
CreateCustomizedObject Method (Page 1825)
How to Edit a Customized Object with VBA (Page 1683)
How to Edit the Group Objects Using VBA (Page 1676)
Group Objects (Page 1674)
Editing Objects with VBA (Page 1660)

How to Edit a Customized Object with VBA

Introduction

Here you will find the following instructions for editing a customized object with VBA:

- Creating a customized object from selected objects
- Ungrouping Customized Objects
- Deleting a customized object

Note

It is not possible to configure a customized object with VBA.
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor").

2. Open the document "ThisDocument" in the Project Explorer:

3. To create a customized object from selected objects, you can for example insert a "CreateCustomizedObject()" procedure in the document "ThisDocument". In this example the customized object "My Customized Object" is created from selected objects:

```vba
Sub CreateCustomizedObject()
' VBA54
Dim objCustomizedObject As HMIResizableObject
Dim objCircle As HMIcircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMIcircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
MsgBox "Objects selected!"
Set objCustomizedObject = ActiveDocument.Selection.CreateCustomizedObject
  '*** The "Configurationdialog" started. ***
  '*** Configure the customize-object with the "configurationdialog"
  '***
  objCustomizedObject.ObjectName = "My Customized Object"
End Sub
```
4. To delete an object, you can for example insert a "DeleteObject()" procedure in the document "ThisDocument". In this example the customized object "My Customized Object" created beforehand is deleted again:
Sub DestroyCustomizedObject()
  'VBA55
  Dim objCustomizedObject As HMICustomizedObject
  Set objCustomizedObject = ActiveDocument.HMIObjects("My Customized Object")
  objCustomizedObject.Destroy
End Sub

5. To delete a customized object, you can for example insert a "DeleteCustomizedObject()" procedure in the document "ThisDocument". In this example the customized object "My Customized Object" created beforehand is deleted:
Sub DeleteCustomizedObject()
  'VBA56
  Dim objCustomizedObject As HMICustomizedObject
  Set objCustomizedObject = ActiveDocument.HMIObjects("My Customized Object")
  objCustomizedObject.Delete
End Sub

6. Always start the procedure with <F5>.

See also

Destroy Method (Page 1835)
Delete Method (Page 1831)
CreateCustomizedObject Method (Page 1825)
Customized Objects (Page 1682)
3.3.5 Creating Dynamics with VBA

3.3.5.1 Creating Dynamics with VBA

Introduction

VBA allows you to add dynamics to properties of pictures and objects and to configure event-controlled actions. VBA provides you with the ActionDynamic object for this purpose:

![Diagram showing the relationship between different objects and their properties]

Legend:
- Yellow: Object and Collection
- Cyan: Object
- Arrow: Transfer: Object depends on type

The ActionDynamic object represents an interface that is dependent on the object type:

- When you configure a dynamic for a property (Property object), the ActionDynamic object inherits the properties of the ScriptInfo, Trigger and DynamicDialog objects.
- When you configure an event-controlled action (Event object), the ActionDynamic object inherits the properties of the ScriptInfo and DirectConnection objects.
Adding dynamics to properties of pictures and objects

VBA enables you to add dynamics to properties of pictures and objects. You can use tags, scripts or the Dynamic dialog to add dynamics. Using dynamics enables you for example to configure a color change for an object in runtime when the value of a variable changes.

Configuring event-controlled actions

You can configure event-controlled actions with VBA. An action (script or direct connection) is triggered then the defined event occurs in runtime. An event may be a change to an object property, for example, or the clicking of a button.

Editing Triggers

You can edit triggers with VBA. Triggers are required when you use dynamics. They determine when a dynamic value is updated in runtime. This may occur at regular intervals, for example, or in the event of a picture change.

When you configure event-controlled actions, the event is the trigger.

See also

Editing Triggers (Page 1707)
Configuring Event-Driven Actions with VBA (Page 1699)
Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)

3.3.5.2 Configuring Dynamics in the Properties of Pictures and Objects

Configuring Dynamics in the Properties of Pictures and Objects

Introduction

VBA enables you to add dynamics to properties of pictures and objects. Dynamic object properties can be changed as a function of a variable value in Runtime, for example. The following methods of adding dynamics are possible:

- Tag connection
- Dynamic dialog
- Scripts
Principle

The following example illustrates the principle of the procedure for adding dynamics to an object property:

```vba
Sub CreateDynamicOnProperty()
    'VBA57
    Dim objVariableTrigger As HMIVariableTrigger
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")
    'Create dynamic with type "direct Variableconnection" at the
    'property "Radius":
    Set objVariableTrigger = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVariableDirect, 
        "NewDynamic1")
    'To complete dynamic, e.g. define cycle:
    With objVariableTrigger
        .CycleType = hmiVariableCycleType_2s
    End With
End Sub
```

Note

Note that a variable is not created simply by specifying the variable name. Use the variable selection dialog to create the variable.

Tag connection

Use the VariableTrigger object to add a dynamic to a property with a direct or indirect variable connection. The dynamic property then responds in runtime to a change in value to the specified variable. To allow this, in VBA you need to specify the variable name (VarName property) and the cycle (CycleTime property).

Dynamic dialog

Use the DynamicDialog object to add a dynamic to a property with the aid of the Dynamic dialog. The dynamic property responds in runtime to a variable's value ranges. The following objects are available for specifying the value range:

- **AnalogResultInfos-Objekt**: Use this object to assign a fixed value to value ranges of a variable or a script. The fixed value is assigned to the dynamic property when the variable value or return value of the script is within the specified value range.
- **BinaryResultInfo Object**: Use this object to assign a fixed value to binary value ranges (zero and non-zero) of a variable or a script. The fixed value is assigned to the dynamic property when the variable value or return value of the script returns one of the two values.
- **VariableStateValue Object**: Use this object to assign a fixed value to the state (e.g. "Upper limit exceeded") of a specified variable. The fixed value is then allocated to the dynamic property when the state occurs.
**Scripts**

Use the ScriptInfo object to add a dynamic to a property with a C or VB script. The property with the dynamic reacts to a script in Runtime and is controlled via a trigger. Use the Trigger object for configuring the trigger.

**See also**

- VariableTrigger Object (Page 2099)
- VariableStateValue Object (Page 2096)
- Trigger Object (Page 2087)
- ScriptInfo Object (Page 2058)
- BinaryResultInfo Object (Page 1918)
- AnalogResultInfos Object (Listing) (Page 1908)
- How to dynamize a property with a VB script (Page 1697)
- How to dynamize a property with a C script (Page 1694)
- How to dynamize a property with the Dynamic dialog (Page 1691)
- How to dynamize a property with a tag connection (Page 1689)
- Creating Dynamics with VBA (Page 1686)

**How to dynamize a property with a tag connection**

**Introduction**

Here you will find the following instructions for dynamizing a property with tag connection:

- Dynamizing a property with direct tag connection
- Dynamizing a property with indirect tag connection
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:

```
Sub AddDynamicAsVariableDirectToProperty()
' Create dynamic at property "Top"
Set objVariableTrigger = objCircle.Top.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "Otto")
' define cycle-time
With objVariableTrigger
  .CycleType = hmiVariableCycleType_2s
End With
End Sub
```

3. To dynamize an object property with a direct tag connection, you can for example insert an "AddDynamicAsVariableDirectToProperty()" procedure in the document. "In this example a circle property "Top" will be made dynamic with the aid of the tag Otto:

```
Sub AddDynamicAsVariableDirectToProperty()
' Create dynamic at property "Top"
Set objVariableTrigger = objCircle.Top.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "Otto")
' define cycle-time
With objVariableTrigger
  .CycleType = hmiVariableCycleType_2s
End With
End Sub
```
4. To dynamize an object property with an indirect tag connection, you can for example insert an "AddDynamicAsVariableInDirectToProperty()" procedure in the document. "In this example a circle property Left" will be made dynamic with the aid of the tag "Anton":

```vba
Sub AddDynamicAsVariableIndirectToProperty()
'VBA59
Dim objVariableTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle2", "HMICircle")
'Create dynamic on property "Left":
Set objVariableTrigger = objCircle.Left.CreateDynamic(hmiDynamicCreationTypeVariableIndirect, "Anton")
'Define cycle-time
With objVariableTrigger
  .CycleType = hmiVariableCycleType_2s
End With
End Sub
```

5. Start the procedure with <F5>.

**See also**

- CycleType Property (Page 2209)
- VarName Property (Page 2548)
- VariableTrigger Object (Page 2099)
- CreateDynamic Method (Page 1828)
- How to dynamize a property with a VB script (Page 1697)
- How to dynamize a property with a C script (Page 1694)
- How to dynamize a property with the Dynamic dialog (Page 1691)
- Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)
- Creating Dynamics with VBA (Page 1686)

**How to dynamize a property with the Dynamic dialog**

**Introduction**

You can use the Dynamic dialog to dynamize properties of pictures and objects depending on certain value ranges or variable states. The following value ranges are available for selection:

- Analog
- Binary
- Bit
- Direct
With VBA you specify the type of value range with the ResultType property. These instructions illustrate the addition of dynamics to an object property with analog value ranges. Additional information dynamization with the dynamic dialog is provided under "DynamicDialog object" in the VBA reference in this documentation.

**Procedure**

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:

   ![Project Explorer Screenshot](image)

3. To Dynamize an object property with the Dynamic dialog, you can for example insert an "AddDynamicDialogToCircleRadiusTypeAnalog()" procedure in the document. In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and three analog value ranges will be created:

4. Start the procedure with <F5>.
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA60
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  'Create dynamic
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "NewDynamic1")
  'Configure dynamic. "ResultType" defines the type of valuerange:
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.Add 50, 40
    .AnalogResultInfos.Add 100, 80
    .AnalogResultInfos.ElseCase = 100
  End With
End Sub

New VBA method to configure dynamization using the Dynamic Dialog

For optimization reasons, an additional new method has been provided:
  • CreateDynamicDialog([Code as String], iResultType as Long) as HMIActionDynamic
The parameter "IResultType" has the following constants:
  • hmiResultTypeDirect = 0
  • hmiResultTypeAnalog = 1
  • hmiResultTypeBool = 2
  • hmiResultTypeBit = 3
In the following example the radius of a circle is given dynamics with the A tag name and a "ResultType" are assigned to the dynamic dialog.

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA820
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  'Create Object
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle","HMICircle")
  'Create dynamic (Tag "myTest" must exist")
  Set objDynDialog = objCircle.Radius.CreateDynamicDialog("myTest",0)
End Sub
Initializing a string property

A string property must be initialized before being made dynamic by assigning a text to it. In the following ToolTipText example, this is done in "objCircle.ToolTipText = "Text".

Sub Dyn()
  'VBA823
  Dim objCircle As HMICircle
  Dim doc As Document
  Dim objDynDialog As HMIDynamicDialog
  Set doc = ActiveDocument
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle","HMICircle")
  objCircle.ObjectName = "Circle1"
  objCircle.BorderColor = RGB(255, 0, 0)
  objCircle.BackColor = RGB(0, 255, 0)
  objCircle.ToolTipText = "Text"
  Set objDynDialog = objCircle.ToolTipText.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'Var'")
End Sub

See also

How to dynamize a property with a tag connection (Page 1689)
ResultType Property (Page 2399)
DynamicDialog Object (Page 1955)
CreateDynamic Method (Page 1828)
How to dynamize a property with a VB script (Page 1697)
How to dynamize a property with a C script (Page 1694)
Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)
Creating Dynamics with VBA (Page 1686)
ToolTipText Property (Page 2445)

How to dynamize a property with a C script

Introduction

When you dynamize a property with a C script, you can assign the C code to the "SourceCode" property. The C script is compiled in the background. The "Compiled" property returns "True" when the C code has been successfully compiled.
Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:
3. To add dynamics to an object property with a C script, you can for example insert an "AddDynamicAsCScriptToProperty()" procedure in the document. In this example the height of a circle is increased by 5 pixels every two seconds in runtime:

```vba
Sub AddDynamicAsCScriptToProperty()
    'VBA61
    Dim objCScript As HMIScriptInfo
    Dim objCircle As HMICircle
    Dim strCode As String
    strCode = "long lHeight;" & vbCrLf & "int check;" & vbCrLf
    strCode = strCode & "GetHeight(""events.PDL",""myCircle");" & vbCrLf
    strCode = strCode & "lHeight = lHeight+5;" & vbCrLf
    strCode = strCode & "check = SetHeight(""events.PDL",""myCircle",""myCircle",""lHeight");" & vbCrLf
    strCode = strCode & "//Return-Type: BOOL" & vbCrLf
    strCode = strCode & "return check;"
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMICircle")
    'Create dynamic for Property "Height":
    Set objCScript = objCircle.Height.CreateDynamic(hmiDynamicCreationTypeCScript)
    'set Sourcecode and cycletime:
    With objCScript
        .SourceCode = strCode
        .Trigger.Type = hmiTriggerTypeStandardCycle
        .Trigger.CycleType = hmiCycleType_2s
        .Trigger.Name = "Trigger1"
    End With
End Sub
```

4. Start the procedure with <F5>.

See also

- Trigger Property (Page 2449)
- ScriptType Property (Page 2407)
- SourceCode Property (Page 2424)
- CycleType Property (Page 2209)
- ScriptInfo Object (Page 2058)
- CreateDynamic Method (Page 1828)
- How to dynamize a property with a VB script (Page 1697)
- How to dynamize a property with the Dynamic dialog (Page 1691)
- How to dynamize a property with a tag connection (Page 1689)
- Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)
- Creating Dynamics with VBA (Page 1686)
How to dynamize a property with a VB script

Introduction
When you dynamize a property with a VB script, you can assign the VB code to the "SourceCode" property. The VB script is compiled in the background. The "Compiled" property returns "True" if the VB code is syntactically correct.

Procedure
1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:
3. To add dynamics to an object property with a VB script, you can for example insert an "AddDynamicAsVBScriptToProperty()" procedure in the document. In this example the radius of a circle is increased by 5 pixels every two seconds in Runtime:

```
Sub AddDynamicAsVBScriptToProperty()
    'VBA62
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMICircle
    Dim strCode As String
    strCode = "Dim myCircle" & vbCrLf & "Set myCircle = "
    strCode = strCode & "HMIRuntime.ActiveScreen.ScreenItems(""myCircle""))"
    strCode = strCode & vbCrLf & "myCircle.Radius = myCircle.Radius + 5"
    Set objCircle =
    ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMICircle")
    'Create dynamic of property "Radius":
    Set objVBScript =
    objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    'Set SourceCode and cycletime:
    With objVBScript
        .SourceCode = strCode
        .Trigger.Type = hmiTriggerTypeStandardCycle
        .Trigger.CycleType = hmiCycleType_2s
        .Trigger.Name = "Trigger1"
    End With
End Sub
```

4. Start the procedure with <F5>.

See also

- How to dynamize a property with a C script (Page 1694)
- Trigger Property (Page 2449)
- SourceCode Property (Page 2424)
- CycleType Property (Page 2209)
- ScriptInfo Object (Page 2058)
- CreateDynamic Method (Page 1828)
- How to dynamize a property with the Dynamic dialog (Page 1691)
- How to dynamize a property with a tag connection (Page 1689)
- Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)
- Creating Dynamics with VBA (Page 1686)
3.3.5.3 Configuring Event-Driven Actions with VBA

Configuring Event-Driven Actions with VBA

Introduction

With VBA you can configure actions for pictures and objects which are triggered when predefined events occur. For example, when the mouse is clicked on an object in Runtime a C script is called whose return value is used for the dynamics of an object property. The following methods of adding dynamics are possible:

- Direct connection
- Scripts

The events that are used for configuring event-controlled actions occur only in Runtime and have nothing to do with the VBA event handlers.

General Procedure

You use the Events property for configuring event-controlled actions with VBA. The way this property is used depends on whether you are configuring an action on an object or picture or a property.

Configuring an action on an object or picture

An action that you configure on a picture or object is triggered when a predefined event occurs, for example when the object is clicked on with the mouse. You configure an action on an object with VBA by using the “Events(Index)” property, where “Index” stands for the triggering event:

```vba
Sub AddActionToObjectTypeCScript()
  'VBA63
  Dim objEvent As HMIEvent
  Dim objCScript As HMIScriptInfo
  Dim objCircle As HMICircle
  'Create circle. Click on object executes an C-action
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
  Set objEvent = objCircle.Events(1)
  Set objCScript = objEvent.Actions.AddAction(hmiActionCreationTypeCScript)
  'Assign a corresponding custom-function to the property "SourceCode":
  objCScript.SourceCode = ""
End Sub
```
Configuring an action on a property
An action that you configure on a property of a picture or object is triggered when the property value changes. You configure an action on a property with VBA by using the "Events(1)" property, where the index "1" stands for the event "Upon change":

```vba
Sub AddActionToPropertyTypeCScript()
  'VBA64
  Dim objEvent As HMIEvent
  Dim objCScript As HMIScriptInfo
  Dim objCircle As HMICircle
  'Create circle. Changing of the Property
  "Radius" should be activate C-Aktion:
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
  Set objEvent = objCircle.Radius.Events(1)
  Set objCScript = objEvent.Actions.AddAction(hmiActionCreationTypeCScript)

  'Assign a corresponding custom-function to the property "SourceCode":
  objCScript.SourceCode = ""
End Sub
```

Direct connection
Use the DirectConnection object to configure a direct connection.

Scripts
Use the ScriptInfo object if you want an event to trigger a C or VB action.

See also
How to configure a VB action with VBA on an event (Page 1705)
Events Property (Page 2223)
ScriptInfo Object (Page 2058)
Event Object (Page 1966)
How to configure a C action with VBA on an event (Page 1703)
How to configure a direct connection with VBA (Page 1701)
Event Handling (Page 1709)
Creating Dynamics with VBA (Page 1686)
How to configure a direct connection with VBA

Introduction

These instructions show you how to configure a direct connection on the basis of two object properties. Further information on the configuring of direct connections with VBA is given in the VBA reference in this documentation under "AutomationName property" and "ObjectName property".

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")
2. In Project Explorer, open the document in which you want to write the VBA code:
3. To configure a direct connection to an object property, you can for example insert an "AddDirectConnectionToObject()" procedure in the document. In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

```
Sub DirectConnection()
  'VBA65
  Dim objButton As HMIButton
  Dim objRectangleA As HMIRectangle
  Dim objRectangleB As HMIRectangle
  Dim objEvent As HMIEvent
  Dim objDConnection As HMI DirectConnection

  'Create objects:
  Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
  Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")

  With objRectangleA
    .Top = 100
    .Left = 100
  End With

  With objRectangleB
    .Top = 250
    .Left = 400
    .BackColor = RGB(255, 0, 0)
  End With

  With objButton
    .Top = 10
    .Left = 10
    .Text = "SetPosition"
  End With

  'Direct connection is initiated by mouseclick:
  Set objDConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)

  With objDConnection
    'Source object: Property "Top" of Rectangle_A
    .SourceLink.Type = hmiSourceTypeProperty
    .SourceLink.ObjectName = "Rectangle_A"
    .SourceLink.AutomationName = "Top"

    'Destination object: Property "Left" of Rectangle_B
    .DestinationLink.Type = hmiDestTypeProperty
    .DestinationLink.ObjectName = "Rectangle_B"
    .DestinationLink.AutomationName = "Left"
  End With
End Sub
```
End Sub

4. Start the procedure with <F5>.

See also

ObjectName Property (Page 2347)
AutomationName Property (Page 2124)
SourceLink Object (Page 2065)
DestLink Object (Page 1945)
DirectConnection Object (Page 1947)
Configuring Event-Driven Actions with VBA (Page 1699)

How to configure a C action with VBA on an event

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:
3. To configure a C action on an event with VBA, you can for example insert a "CreateCActionToClickedEvent()" procedure in the document. In this example a button and a circle will be inserted in the active picture. In Runtime the height increases every time you click the button:

```
Sub CreateCActionToClickedEvent()
'VBA66
Dim objButton As HMIButton
Dim objCircle As HMICircle
Dim objEvent As HMIEvent
Dim objCScript As HMIScriptInfo
Dim strCode As String
strCode = "long lHeight;" & vbCrLf & "int check;" & vbCrLf & "lHeight = GetHeight (""events.PDL"", ""myCircle"");"
strCode = strCode & vbCrLf & "lHeight = lHeight+5;" & vbCrLf & "check = "
strCode = strCode & vbCrLf & "SetHeight(""events.PDL"", ""myCircle"", lHeight);"
strCode = strCode & vbCrLf & "//Return-Type: Void"
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircle
    .Top = 100
    .Left = 100
    .BackColor = RGB(255, 0, 0)
End With
With objButton
    .Top = 10
    .Left = 10
    .Text = "Increase height"
End With
'Configure directconnection:
Set objCScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeCScript)
With objCScript
    .SourceCode = strCode
End With
End Sub
```

4. Start the procedure with <F5>.

See also

- ScriptInfo Object (Page 2058)
- Events Object (Listing) (Page 1967)
How to configure a VB action with VBA on an event

Procedure

1. Open the VBA editor in Graphics Designer (<ALT+F11> or "Tools" > "Macros" > "Visual Basic Editor")

2. In Project Explorer, open the document in which you want to write the VBA code:
3. To configure an event-oriented VB action with VBA, you can for example insert a "CreateVBActionToClickedEvent()" procedure in the document. In this example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle enlarges every time you click the button:

```vba
Sub CreateVBActionToClickedEvent()
'VBA67
Dim objButton As HMIButton
Dim objCircle As HMICircle
Dim objEvent As HMIEvent
Dim objVBScript As HMIScriptInfo
Dim strCode As String
strCode = "Dim myCircle" & vbCrLf & "Set myCircle = "
strCode = strCode & "HMIRuntime.ActiveScreen.ScreenItems(""Circle_VB"")"
strCode = strCode & vbCrLf & "myCircle.Radius = myCircle.Radius + 5"
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VB", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircle
  .Top = 100
  .Left = 100
  .BackColor = RGB(255, 0, 0)
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 120
  .Text = "Increase Radius"
End With
'Define event and assign sourcecode:
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScrip t)
With objVBScript
  .SourceCode = strCode
End With
End Sub
```

4. Start the procedure with <F5>.

See also

Actions Object (Listing) (Page 1898)
ScriptInfo Object (Page 2058)
Events Object (Listing) (Page 1967)
Configuring Event-Driven Actions with VBA (Page 1699)
3.3.5.4 Editing Triggers

Introduction

You use triggers in connection with dynamics for graphics objects and for triggering actions on object properties. Examples of triggers include:

- Tags: When the value of a tag is changed or its limit is exceeded at either extreme
- Standard cycle: Cyclic execution of the action. The length of the cycles is selectable between 250 ms and 1 h. In addition, you can also use customized cycles that you define yourself.
- Picture cycle: A cyclic trigger is used as the trigger. This cycle provides the option of defining the cycles of all the actions, tag connections and dynamic dialogs used in a picture centrally.
- Window Cycle: A cyclic trigger is used as the trigger. This values applies to all actions, tag links and dynamic dialogs, which were configured with the trigger type "Window cycle".

When you configure an action that responds to an event on a graphics object, the triggering event is the trigger.

Configuring triggers with VBA

Use the Trigger object to configure a trigger with VBA. If you intend to use a variable as the trigger, use the VariableTrigger object:

You determine the type of trigger with the Type property. Use the VariableTriggers property when you configure a variable as the trigger.

See also

Examples of Editing Triggers with VBA (Page 1708)
VariableTrigger Object (Page 2099)
Trigger Object (Page 2087)
ScriptInfo Object (Page 2058)
Examples of Editing Triggers with VBA

Introduction

The four examples below illustrate how you can create the following triggers with VBA:

- Standard cycle
- Tag
- Picture cycle
- Window Cycle

In all of these examples a circle is inserted into the active picture, with the radius of the circle being dynamized with a VB action.

The procedure for adding dynamics to a property with variable connection is explained under "Adding dynamics to a property with a variable connection" in this documentation.

Example 1: Standard cycle

Sub DynamicWithStandardCycle()
    'VBA68
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_Standard", "HMICircle")
    Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    With objVBScript
        .Trigger.Type = hmiTriggerTypeStandardCycle
        "CycleType"-specification is necessary:
        .Trigger.CycleType = hmiCycleType_10s
        .Trigger.Name = "VBA_StandardCycle"
        .SourceCode = ""
    End With
End Sub

Example 2: Tag

Sub DynamicWithVariableTriggerCycle()
    'VBA69
    Dim objVBScript As HMIScriptInfo
    Dim objVarTrigger As HMIVariableTrigger
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VariableTrigger", "HMICircle")
    Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    With objVBScript
        Set objVarTrigger = .Trigger.VariableTriggers.Add("VarTrigger", hmiVariableCycleType_10s)
        .SourceCode = ""
    End With
End Sub
Example 3: Picture cycle

Sub DynamicWithPictureCycle()
  'VBA70
  Dim objVBScript As HMIScriptInfo
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_Picture", "HMICircle")
  Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
  With objVBScript
    .Trigger.Type = hmiTriggerTypePictureCycle
    .Trigger.Name = "VBA_PictureCycle"
    .SourceCode = ""
  End With
End Sub

Example 4: Window Cycle

Sub DynamicWithWindowCycle()
  'VBA71
  Dim objVBScript As HMIScriptInfo
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_Window", "HMICircle")
  Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
  With objVBScript
    .Trigger.Type = hmiTriggerTypeWindowCycle
    .Trigger.Name = "VBA_WindowCycle"
    .SourceCode = ""
  End With
End Sub

See also

VariableTrigger Object (Page 2099)
Trigger Object (Page 2087)
ScriptInfo Object (Page 2058)
Editing Triggers (Page 1707)

3.3.6 Event Handling

Introduction

In the Graphics Designer, events occur when certain actions are taken (for example when a picture is opened). You can respond to an event with a predefined VBA event handler in order to execute instructions.
The events occur only during configuring in the Graphics Designer and are not available in Runtime. These events must not be confused with the events (e.g. mouse click, property change) occurring on graphic objects and pictures.

**Note**
When the Graphics Designer is open, events are also triggered by other editors. This applies, for example, to the modification of picture properties in WinCCExplorer. Close the Graphics Designer when you are making changes to pictures in other editors. This prevents events from being executed when you do not want them to be.

**Note**
If you open a picture in Graphics Designer, not only the "DocumentOpened event" of the active picture but also that of the "Project Template" and of the "Global Template" are triggered. The VBA code of the "DocumentOpened event" is thereby executed twice.

You must intercept this behavior with the event handler.

### General Procedure

In event handling there are events with and without forwarding. You can recognize an event with forwarding by the presence of the "CancelForwarding" parameter. An event without forwarding does not have this parameter. When an event occurs, it is sent to the active picture and then forwarded to the "Global Template".

![Diagram showing event flow](image)

An event with forwarding is therefore forwarded by default via the document "Project Template" to the document "Global Template".

#### Preventing forwarding

You can prevent the forwarding of an event by setting the "CancelForwarding" parameter to "True" in the VBA event handler:

```vba
Sub Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
    'VBA72
    CancelForwarding = True
    MsgBox "Object's property has been changed!"
End Sub
```

### Picture-specific and application-specific events

Quite apart from the information given above about events with and without forwarding, the Graphics Designer differentiates between picture-specific and application-specific events:
Picture-specific events

Picture-specific events always respond to actions that occur in the active picture in the Graphics Designer. Such actions include, for example, the changing of object properties or saving the active picture. You can obtain a list of available picture-specific events by choosing "Document" in the VBA editor:

```
Private Sub Document_HMIObjectPropertyChanged()
    Dim objProperty As HMIProperty
```

Application-specific events

Application-specific events respond to actions that occur in the "Graphics Designer" application. Such actions include, for example, starting the Graphics Designer or creating an object in the component library.

To make the application-specific events available, write the following statement in the VBA editor at the start of the document (preferably the "Project Template" or "Global Template"):

```
Dim WithEvents <Name> As grafexe.Application
```

The effect of this statement is that it will now also be possible to select the application-specific events from the list in the Graphics Designer:

```
Private Sub objGDAplication_Started()
    MsgBox ("Graphics Designer has been started")
End Sub
```
Example 1: Occurrence of an event with forwarding

The illustration shows the sequence that follows from the occurrence of an event with forwarding:

Note

There are events which are both picture-specific and application-specific (for example BeforeDocumentSave). When such an event occurs, the program checks whether the corresponding application-specific event handler has been defined. The sequence shown above does not begin until after that.

Example 2: Occurrence of an event without forwarding

The illustration shows the sequence that follows from the occurrence of an event without forwarding:
Disabling event handling

You can disable event handling by setting the "DisableVBAEvents" property for the Application object to "True".

See also

DisableVBAEvents Property (Page 2214)
Organizing VBA Code in a WinCC Project (Page 1615)

3.3.7 Accessing External Applications with VBA

3.3.7.1 Accessing External Applications with VBA

Introduction

You can use VBA to access programs which support VBA, for example products in the Microsoft Office family. This enables you, for example, to read out values from an MS Excel worksheet and then assign these to object properties.

Note

No direct usage of Unicode in Excel VBA and Word VBA

Excel VBA and Word VBA supports the use of Unicode characters only by means of <ChrW(unicode-id) function.
Registering an external application

You have to integrate an external application in the VBA editor in order to make its object library available. To do this, select the "References" option in the "Tools" menu in the VBA editor. In the "References" dialog you can then select the required object library:

![References dialog]

Note

You must attach the external application to all projects which you want to be able to access the external application; do this in the VBA editor Project Explorer.

See also

Example: Accessing MS Excel with VBA (Page 1714)

3.3.7.2 Example: Accessing MS Excel with VBA

Introduction

The following three examples illustrate how to access MS Excel. For these examples to work, the MS Excel object library must be integrated via a reference.
Note

No direct usage of Unicode in Excel VBA and Word VBA

Excel VBA and Word VBA supports the use of Unicode characters only by means of \texttt{ChrW(unicode-id)} function.
Example 1

In this example the default object list of the Graphics Designer is exported in an Excel worksheet. The object properties are taken into account, as is the question of whether dynamics can be used with these properties. The VBA data type is also displayed.
Sub ExportDefObjListToXLS()
'VBA73
'Microsoft Excel Object Library needs to be referenced
Dim objGDApplication As grafexe.Application
Dim objHMIObject As grafexe.HMIObject
Dim objProperty As grafexe.HMIProperty
Dim objXLS As Excel.Application
Dim objWSheet As Excel.Worksheet
Dim objWBook As Excel.Workbook
Dim rngSelection As Excel.Range
Dim lRow As Long
Dim lRowGroupStart As Long

'define local errorhandler
On Local Error GoTo LocErrTrap

'Set references to the applications Excel and GraphicsDesigner
Set objGDApplication = grafexe.Application
Set objXLS = New Excel.Application

'Create workbook
Set objWBook = objXLS.Workbooks.Add()
objWBook.SaveAs objGDApplication.ApplicationDataPath & "DefaultObjekte.xls"

'Add's new worksheet to the new workbook
Set objWSheet = objWBook.Worksheets.Add
objWSheet.Name = "DefaultObjekte"
lRow = 1

'Every object of the DefaultHMIObjects-collection will be written to the worksheet with their objectproperties.
'For better overview the objects will be grouped.
For Each objHMIObject In objGDApplication.DefaultHMIObjects
    DoEvents
    objWSheet.Cells(lRow, 1).value = objHMIObject.ObjectName
    objWSheet.Cells(lRow, 2).value = objHMIObject.Type
    lRow = lRow + 1
    lRowGroupStart = lRow
    For Each objProperty In objHMIObject.Properties
        'Write displayed name and automationname of property into the worksheet
        objWSheet.Cells(lRow, 2).value = objProperty.DisplayName
        objWSheet.Cells(lRow, 3).value = objProperty.Name
        'Write the value of property, datatype and if their dynamicable into the worksheet
        If Not IsEmpty(objProperty.value) Then _
            objWSheet.Cells(lRow, 4).value = objProperty.value
            objWSheet.Cells(lRow, 5).value = objProperty.IsDynamicable
            objWSheet.Cells(lRow, 6).value = TypeName(objProperty.value)
            objWSheet.Cells(lRow, 7).value = VarType(objProperty.value)
            lRow = lRow + 1
    Next objProperty
Next objHMIObject

'Select and groups the range of object-properties in the worksheet
Set rngSelection = objWSheet.Range(objWSheet.Rows(lRowGroupStart), _
    objWSheet.Rows(lRow - 1))

LocErrTrap:
On Local Error Resume Next
End Sub
rngSelection.Select
rngSelection.Group
Set rngSelection = Nothing
' Insert empty row
lRow = lRow + 1
Next objHMIObject
objWSheet.Columns.AutoFit
Set objWSheet = Nothing
objWBook.Save
objWBook.Close
Set objWBook = Nothing
objXLS.Quit
Set objXLS = Nothing
Set objGDApplication = Nothing
Exit Sub
LocErrTrap:
MsgBox Err.Description, , Err.Source
Resume Next
End Sub
Example 2

In this example all objects of the active picture are exported to an Excel worksheet. The properties taken into account are Position X, Position Y, Width, Height and Layer:
Sub ExportObjectListToXLS()
'VBA74
Dim objGDApplication As grafexe.Application
Dim objDoc As grafexe.Document
Dim objHMIObject As grafexe.HMIObject
Dim objHMIProperty As grafexe.HMIProperty
Dim objXLS As Excel.Application
Dim objWSheet As Excel.Worksheet
Dim objWBook As Excel.Workbook
Dim lRow As Long

'Define local errorhandler
On Local Error GoTo LocErrTrap

'Set references on the applications Excel and GraphicsDesigner
Set objGDApplication = grafexe.Application
Set objDoc = objGDApplication.ActiveDocument
Set objXLS = New Excel.Application

'Create workbook
Set objWBook = objXLS.Workbooks.Add()
objWBook.SaveAs objGDApplication.ApplicationDataPath & "Export.xls"

'Create worksheet in the new workbook and write headline
'The name of the worksheet is equivalent to the documents name
Set objWSheet = objWBook.Worksheets.Add
objWSheet.Name = objDoc.Name
objWSheet.Cells(1, 1) = "Objektname"
objWSheet.Cells(1, 2) = "Objekttyp"
objWSheet.Cells(1, 3) = "ProgID"
objWSheet.Cells(1, 4) = "Position X"
objWSheet.Cells(1, 5) = "Position Y"
objWSheet.Cells(1, 6) = "Width"
objWSheet.Cells(1, 7) = "Höhe"
objWSheet.Cells(1, 8) = "Ebene"
lRow = 3

'Every objects will be written with their objectproperties width, height, pos x, pos y and layer to Excel. If the object is an 'ActiveX-Control the ProgID will be also exported.
For Each objHMIObject In objDoc.HMIObjects
    DoEvents
    objWSheet.Cells(lRow, 1).value = objHMIObject.ObjectName
    objWSheet.Cells(lRow, 2).value = objHMIObject.Type
    If UCase(objHMIObject.Type) = "HMIACTIVEXCONTROL" Then
        objWSheet.Cells(lRow, 3).value = objHMIObject.ProgID
    End If
    objWSheet.Cells(lRow, 4).value = objHMIObject.Left
    objWSheet.Cells(lRow, 5).value = objHMIObject.Top
    objWSheet.Cells(lRow, 6).value = objHMIObject.Width
    objWSheet.Cells(lRow, 7).value = objHMIObject.Height
    objWSheet.Cells(lRow, 8).value = objHMIObject.Layer
    lRow = lRow + 1
Next objHMIObject
objWSheet.Columns.AutoFit
 LocErrTrap: On Error Resume Next
Set objWSheet = Nothing
objWBook.Save
objWBook.Close
Set objWBook = Nothing
objXLS.Quit
Set objXLS = Nothing
Set objDoc = Nothing
Set objGDApplication = Nothing
Exit Sub
LocErrTrap:
MsgBox Err.Description, , Err.Source
Resume Next
End Sub
Example 3

In this example objects are imported from the Excel worksheet created in example 2. The properties taken into account are Position X, Position Y, Width, Height and Layer:
Sub ImportObjectListFromXLS()
'VBA75
Dim objGDApplication As grafxe.Application
Dim objDoc As grafxe.Document
Dim objHMIObject As grafxe.HMIOBJECT
Dim objXLS As Excel.Application
Dim objWSheet As Excel.Worksheet
Dim objWBook As Excel.Workbook
Dim lRow As Long
Dim strWorkbookName As String
Dim strWorksheetName As String
Dim strSheets As String

'define local errorhandler
On Local Error GoTo LocErrTrap

'Set references on the applications Excel and GraphicsDesigner
Set objGDApplication = Application
Set objDoc = objGDApplication.ActiveDocument
Set objXLS = New Excel.Application

'Open workbook. The workbook have to be in datapath of GraphicsDesigner
strWorkbookName = InputBox("Name of workbook: ", "Import of objects")
Set objWBook = objXLS.Workbooks.Open(objGDApplication.ApplicationDataPath &
strWorkbookName)
If objWBook Is Nothing Then
MsgBox "Open workbook fails!" & vbCrLf & "This function is canceled!", vbCritical, "Import od objects"
Set objDoc = Nothing
Set objGDApplication = Nothing
Set objXLS = Nothing
Exit Sub
End If

'Read out the names of all worksheets contained in the workbook
For Each objWSheet In objWBook.Sheets
strSheets = strSheets & objWSheet.Name & vbCrLf
Next objWSheet
strWorksheetName = InputBox("Name of table to import: ", "Import of objects")
Set objWSheet = objWBook.Sheets(strWorksheetName)
lRow = 3

'Import the worksheet as long as in actual row the first column is empty.
'Add with the outreaded data new objects to the active document and
'assign the values to the objectproperties
With objWSheet
While (.Cells(lRow, 1).value <> vbNullString) And (Not IsEmpty(.Cells(lRow, 1).value))
'Add the objects to the document as its objecttype,
'do nothing by groups, their have to create before.
If (UCase(.Cells(lRow, 2).value) = "HMIGROUP") Then
Else
If (UCase(.Cells(lRow, 2).value) = "HMIACTIVEXCONTROL") Then
Set objHMIObject = objDoc.HMIOBJECTs.AddActiveXControl(.Cells(lRow, 1).value, .Cells(lRow, 3).value)
End If
End If
Next lRow
End With

LocErrTrap:
Set objWBook = Nothing
Set objWSheet = Nothing
Set objDoc = Nothing
Set objGDApplication = Nothing
Set objXLS = Nothing
End Sub
Else
    Set objHMIObject = objDoc.HMIObjets.AddHMIObject(.Cells(lRow, 1).value, .Cells(lRow, 2).value)
    objHMIObject.Left = .Cells(lRow, 4).value
    objHMIObject.Top = .Cells(lRow, 5).value
    objHMIObject.Width = .Cells(lRow, 6).value
    objHMIObject.Height = .Cells(lRow, 7).value
    objHMIObject.Layer = .Cells(lRow, 8).value
End If
Set objHMIObject = Nothing
lRow = lRow + 1
Wend
End With
objWBook.Close
Set objWBook = Nothing
objXLS.Quit
Set objXLS = Nothing
Set objDoc = Nothing
Set objGDApplication = Nothing
Exit Sub
LocErrTrap:
MsgBox Err.Description, , Err.Source
Resume Next
End Sub
3.4 VBA in the WinCC Configuration Studio

3.4.1 VBA in the WinCC Configuration Studio

As of WinCC V7.4 the WinCC Configuration Studio supports the VBA interface.

With VBA, you can access all editors in the WinCC Configuration Studio, for example, to create, change, or delete data. VBA access is also supported for editors of WinCC options that are integrated in the WinCC Configuration Studio.

You open the VBA editor in the WinCC Configuration Studio with "Tools > Visual Basic Editor".

VBA in the WinCC editors

The WinCC editors, objects and data records in the WinCC Configuration Studio can be selected either by name or by index.

Note

Notes on VBA scripts in the WinCC Configuration Studio

Note the following when creating VBA scripts:

- To access columns and tabs, note the respective WinCC user interface language.
- Hidden columns can result in incorrect index assignments when editing via scripts.
- A sequential index over all hierarchical levels is created with an enumeration.

NOTICE

Deleting objects

If you delete individual objects in the Configuration Studio with VBA scripts, associated objects are deleted as well.

Examples:

- Message classes and message types including within
- Tag groups and contained tags

Important functions:

<table>
<thead>
<tr>
<th>DataGrid.Tabs</th>
<th>Enumerate tabs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataGrid.UsedRange</td>
<td>Detect number of data records or rows used</td>
</tr>
<tr>
<td>DataGrid.Rows.Name</td>
<td>Detect object type in a DataGrid, e.g. &quot;Tag&quot;</td>
</tr>
</tbody>
</table>

The following script examples introduce other important functions.
Show hidden columns

To avoid incorrect index assignments, expand the hidden columns.

```vba
Sub ShowAllColumns()
    For i = 1 To DataGrid.EntireColumns.Count
        DataGrid.EntireColumns(i).Visible = True
    Next i
End Sub
```

Example "Tag Management, User Administrator, Graphics Designer"

You can find a detailed example under "Example: VBA in the WinCC Configuration Studio (Page 1726)".

The example contains the following functions:

- Change the user interface language
  To run a script for a specific language, set the user interface language at the beginning of the script.

- Select editor in the WinCC Configuration Studio

- Navigation area: Select tree node

- Data area: Use of row names to detect the object type
  To ensure that the script accesses the correct tab, for example, check the "Name" property of the desired row.

- Data area: Select tab

- Data area: Create new data record
  You determine the row for the new data record and write the values in the cells of this row.

- Data area: Change value

- Data area: Delete entry
  To delete a tag in the data area, for example, check if the tag exists and then delete the entire row.
  When you delete an object, all associated objects are deleted. If you delete a tag group, for example, all tags of the group are deleted.

- Create objects in the Graphics Designer
  To access the class "HMIGO" with VBA, reference the "HMI General Objects 1.0 Type Library" in the VBA script. For each created object, you need in the VBA script to reference the respective block library.

3.4.2 Example: VBA in the WinCC Configuration Studio

This example illustrates the object model used.

The following actions are performed:

<table>
<thead>
<tr>
<th>Example</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddTag</td>
<td>Create a tag in Tag Management</td>
</tr>
<tr>
<td>ModifyTag</td>
<td>Change the start value of a newly created tag</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Example</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveTag</td>
<td>Delete a created tag again</td>
</tr>
<tr>
<td>AddGroup</td>
<td>Create a user group in the User Administrator</td>
</tr>
<tr>
<td>RemoveGroup</td>
<td>Delete a created user group again</td>
</tr>
<tr>
<td>AddUser</td>
<td>Create a user and user group in the User Administrator</td>
</tr>
<tr>
<td>RemoveUser</td>
<td>Delete a created user again</td>
</tr>
<tr>
<td>DemoUsageGraphicsDesigner</td>
<td>Access to the Graphics Designer</td>
</tr>
<tr>
<td></td>
<td>A process picture is created with the &quot;Static text&quot; object.</td>
</tr>
</tbody>
</table>
Example: Tag Management, User Administrator, Graphics Designer

'VBA907
' Version of this sample: V1.0
'
' With WinCC V7.4 CCConfigStudio supports a VBA interface.
' The following examples should help understanding the provided objectmodel.
'
' Warning:
' When deleting objects which are containers/parent objects for other objects (message
classes -> message types, tag groups -> tags, ...),
' all child objects will be deleted, too.
'
' Remarks:
' User Administrator -> Setting password: Not supported due to security issues.

Option Explicit

' This method will run all available sample methods
' Please use a new/empty WinCC project
Sub FullDemo()

    AddTag
    MsgBox "Tag created", vbOKOnly, "VBA-Demo"

    ModifyTag
    MsgBox "Tag modified", vbOKOnly, "VBA-Demo"

    RemoveTag
    MsgBox "Tag removed", vbOKOnly, "VBA-Demo"

    AddGroup
    MsgBox "User Group created", vbOKOnly, "VBA-Demo"

    AddUser
    MsgBox "User created", vbOKOnly, "VBA-Demo"

    RemoveUser
    MsgBox "User removed", vbOKOnly, "VBA-Demo"

    RemoveGroup
    MsgBox "User group removed", vbOKOnly, "VBA-Demo"

    DemoUsageGraphicsDesigner
    MsgBox "Demo finished.", vbOKOnly, "VBA-Demo"

End Sub

' Add a tag in Tag Management
Sub AddTag()

    Dim currLang, i, newRowNo

    ' Save currently selected language
    currLang = ConfigStudio.Language
'Set language to English
ConfigStudio.Language = 1033

'Select editor "Tag Management"
' Valid parameters for specifying editor are names and IDs
ConfigStudio.Editors("Tag Management").Select

'Select treenode "Internal tags"
' Valid parameters for specifying treenodes are names and IDs
NavigationTree.Nodes("Internal tags").Select

'Make sure that the correct tab of DataGrid is selected
'One possibility is to check the object type of the rows object with its name property:
'Property "Name" reflects the object type of the currently selected DataGrid
If DataGrid.Rows.Name = "Tag" Then
   'UsedRange.EntireRow returns the number of used rows in DataGrid.
   newRowNo = DataGrid.UsedRange.EntireRow.Count + 1

   'Add tag via entering values into cells of the first empty row in DataGrid
   ' Valid parameters for specifying columns are names and IDs
   DataGrid.Cells(newRowNo, 1) = "NewTag"
   DataGrid.Cells(newRowNo, "Data type") = "Text tag 16-bit character set"
End If

'Reset language
ConfigStudio.Language = currLang
End Sub

'Changes the start value of the previously created tag
Sub ModifyTag()
    Dim currLang, i

    currLang = ConfigStudio.Language
    ConfigStudio.Language = 1033

    ConfigStudio.Editors("Tag Management").Select
    NavigationTree.Nodes("Internal tags").Select

    If DataGrid.Rows.Name = "Tag" Then
        'Modify tag property via usage of tag name to specify the row
        DataGrid.Cells("NewTag", "Start Value") = "abc"
    End If

    ConfigStudio.Language = currLang
End Sub

'Removes previously created tag
Sub RemoveTag()
    Dim currLang, i
currLang = ConfigStudio.Language
ConfigStudio.Language = 1033

ConfigStudio.Editors("Tag Management").Select
NavigationTree.Nodes("Internal tags").Select

If DataGrid.Rows.Name = "Tag" Then
    'Check if tag tag exists via enumerating rows
    For i = 1 To DataGrid.UsedRange.EntireRow.Count
        If DataGrid.Cells(i, "Name") = "NewTag" Then
            'Select the row first before deleting it
            DataGrid.Rows("NewTag").Select
            'Delete the whole row
            DataGrid.Rows("NewTag").EntireRow.Delete
        End If
    Next
End If

ConfigStudio.Language = currLang

End Sub

'Add a group in User Administrator
Sub AddGroup()

    Dim currLang, i

    currLang = ConfigStudio.Language
    ConfigStudio.Language = 1033

    ConfigStudio.Editors("User Administrator").Select
    NavigationTree.Nodes("User Administrator").Select

    'Add user group / Example for enumeration of tabs
    For i = 1 To DataGrid.Tabs.Count
        If DataGrid.Tabs(i).Name = "Groups" Then
            DataGrid.Tabs(i).Select
            DataGrid.Cells(DataGrid.UsedRange.EntireRow.Count + 1, 1) = "NewGroup"
        End If
    Next

    ConfigStudio.Language = currLang

End Sub

'Delete previously created user group
Sub RemoveGroup()

    Dim currLang, i

    currLang = ConfigStudio.Language
    ConfigStudio.Language = 1033

    ConfigStudio.Editors("User Administrator").Select
NavigationTree.Nodes("User Administrator").Select

For i = 1 To DataGrid.Tabs.Count
    If DataGrid.Tabs(i).Name = "Groups" Then
        DataGrid.Tabs(i).Select
        DataGrid.Rows("NewGroup").Select
        DataGrid.Rows("NewGroup").EntireRow.Delete
    End If
Next i

ConfigStudio.Language = currLang

End Sub

'Add an user and a group in User Administrator
Sub AddUser()
    Dim currLang, i, newRowNo
    currLang = ConfigStudio.Language
    ConfigStudio.Language = 1033
    ConfigStudio.Editors("User Administrator").Select
    NavigationTree.Nodes("User Administrator").Select

    'Add user
    For i = 1 To DataGrid.Tabs.Count
        If DataGrid.Tabs(i).Name = "Users" Then
            DataGrid.Tabs(i).Select
            newRowNo = DataGrid.UsedRange.EntireRow.Count + 1
            DataGrid.Cells(newRowNo, "User Name") = "NewUser"
            'NOT SUPPORTED: DataGrid.Cells(newRowNo, "Password") = "MySuperSavePassword"
            DataGrid.Cells(newRowNo, "WebNavigator") = True
            DataGrid.Cells(newRowNo, "Group name") = "NewGroup"
            Exit For
        End If
    Next i
    ConfigStudio.Language = currLang

End Sub

'Delete previously created user
Sub RemoveUser()
    Dim currLang, i
    currLang = ConfigStudio.Language
    ConfigStudio.Language = 1033
    ConfigStudio.Editors("User Administrator").Select
    NavigationTree.Nodes("User Administrator").Select

    'Remove user
For i = 1 To DataGrid.Tabs.Count
    If DataGrid.Tabs(i) = "Users" Then
        DataGrid.Tabs(i).Select
        DataGrid.Rows("NewUser").Select
        DataGrid.Rows("NewUser").EntireRow.Delete
        Exit For
    End If
Next i

ConfigStudio.Language = currLang
End Sub

'Sample for using VBA in Graphics Designer via CCConfigStudio
Sub DemoUsageGraphicsDesigner()

'Method 1:
'Use this code to create programmatically a reference to Graphics Designer Type Library
    Dim appGD, screen
    Set appGD = CreateObject("grafexe.Application")
'or
'Method 2:
'Add reference to "WinCC Graphics Designer Type Library V1.0" via menu -> Extras -> References
'This will also enable IntelliSense functionality
    'Dim appGD As New grafexe.Application
    'Dim screen As grafexe.Document
    appGD.Visible = True
    Set screen = appGD.Documents.Add(1) 'hmiOpenDocumentTypeVisible

'To be able to add screenitems it is necessary to reference the corresponding type library for each object
    'Dim statTxtObj As HMIStaticText 'add reference "HMI StaticText Object 1.0 Library"
    Dim statTxtObj
    Set statTxtObj = screen.HMIObjects.AddHMIObject("Static Text", "HMIStaticText")
    With statTxtObj
        .Top = 50
        .Left = 50
        .Width = 260
        .Height = 60
        .AlignmentTop = 1
        .AlignmentLeft = 1
        .Text = "Screen created and saved." + vbCrLf + "Please close Graphics Designer manually."
    End With

    screen.SaveAs appGD.ApplicationDataPath + "SampleScreen.pdl"

    Set statTxtObj = Nothing
    Set screen = Nothing
    Set appGD = Nothing
End Sub

See also

VBA in the WinCC Configuration Studio (Page 1725)
3.5 AddIns

3.5.1 AddIns

Introduction

An AddIn is a code which cannot be viewed and is stored as a DLL. Addins can make new functions available by registering the DLL concerned in the operating system and loading it to you as a user, the advantage of addins is that they provide functions which are tailor made for the associated application. For example if you working on different computers during configuration and frequently use VBA macros, you can combine these VBA macros in one or more addins. When you change to a different computer you need only copy the addin and you can then access the customary functions on the new workstation.

As a developer, you can use the Graphics Designer program library in a development environment to create MS Visual Studio 6.0 addins, for example, and protect your code from intrusion.

Addins in the Graphics Designer

In the Graphics Designer you can use all the addins that have been developed for the Graphics Designer and registered in the operating system of the computer you are using for configuration.

You can automatically load an addin when you start the Graphics Designer if you frequently need the functions the addins contain.

When you no longer need the functions of an addin, you can unload it again at any time.

See also

How to Configure an AddIn in the Graphics Designer (Page 1737)
Example: Creating Add Ins (Page 1738)
Linking Add Ins (Page 1734)

3.5.2 Linking Add Ins

Introduction

In the Graphics Designer use the Addin Manager to define the way addins that can be used in the Graphics Designer will behave on loading.
Requirements

- An add-in must be registered in the operating system, e.g. by entering the "regsvr32 filename.dll" command at the input prompt.

- To register VBA addins, "Microsoft Visual Basic for Applications" must be installed. The installation is available in the following ways:
  - Microsoft Office: During installation of Microsoft Office products, for example MS Excel or MS Word, Visual Basic for Applications is automatically installed at the same time.
  - Later installation from Microsoft Office Setup: You can select to only install Visual Basic for Applications in Microsoft Office Setup with the user-defined installation.
  - Download of the VBA Runtime environment: Microsoft offers a download of the VBA Runtime environment at the following links:
    - "VBRun60.exe" file for V6.0: http://support.microsoft.com/kb/192461/ (http://support.microsoft.com/kb/192461/)
    - "VBRun60sp6.exe" file for V6.0 SP6: http://support.microsoft.com/kb/290887/ (http://support.microsoft.com/kb/290887/)

- Make sure that the current file "MSAddndr.DLL" is integrated together with the VB6 add-in DLL. For more detailed information, refer to:
  - http://support.microsoft.com/kb/192136/ (http://support.microsoft.com/kb/192136/)
  - http://support.microsoft.com/kb/2792179/ (http://support.microsoft.com/kb/2792179/)
  - http://support.microsoft.com/kb/957924/ (http://support.microsoft.com/kb/957924/)
Starting AddIn Manager

To start the Addin Manager, go to the Graphics Designer and select the command "Macros > "AddIn Manager".

Automatically Loading an Addin

If the addin contains new functions that you always need in the Graphics Designer, you can load the addin automatically when you open the Graphics Designer.

To do this go to the Addin Manager, select the addin and enable the "Load on Startup" checkbox.

Note

Depending on how the addin is programmed, the function contained in the addin can also be entered in the menu "Tools > Macros > AddIns". You can then start the function just by clicking on it.

Manually Loading or Unloading an Addin

You can also load an addin manually if you need its functions for particular purposes only (such as test routines).

To load or unload an addin manually, go to the Addin Manager, select the addin and enable the "Loaded/Unloaded" checkbox.
See also

How to Configure an AddIn in the Graphics Designer (Page 1737)
Example: Creating Add Ins (Page 1738)
AddIns (Page 1734)

3.5.3 How to Configure an AddIn in the Graphics Designer

Requirements

An add-in must be registered in the operating system, e.g. by entering the "regsvr32 filename.dll" command at the input prompt.

Procedure

1. Start the Graphics Designer and open the project to which you want to link the addin.

2. To call the Addin Manager, select the menu command "Tools" > "AddIn Manager". The Addin Manager opens. The "Available Add-Ins" list shows all the addins that are available, together with their current load status:

3. For each addin define whether it is to be loaded and if so when. To do this select the addin concerned and enable the appropriate checkbox under "Load Behavior".
4. To unload an addin, select the addin concerned and disable the "Load/Unload" checkbox under "Load Behavior".

5. Click OK.

Result

Depending how the addin is programmed, the function contained in the addin is either listed in the "Tools" > "AddIns" menu or reacts to an event handler in the Graphics Designer.

If the addin is started by means of an event handler (e.g. Started Event), the "On Startup" checkbox should be enabled for the addin.

See also

Linking Add Ins (Page 1734)
AddIns (Page 1734)

3.5.4 Example: Creating Add Ins

3.5.4.1 Example: Creating Add Ins

Introduction

In order to create Addins, this documentation contains an example for Visual Basic 6.0, which creates a runnable Addin for use in the Graphic Designer.

Requirements

MS Visual Studio 6.0 must be installed on the configuration computer.

You should have programming experience if you wish to use the sample code as a basis for developing addins of your own.

Example: Program Template for Visual Basic 6.0

Use the event handler "AddInInstance_OnConnection" to generate an instance of the Graphics Designer. In order for the addin to be able to access the Graphics Designer, it is mandatory to declare the application.

See also

Example: Creating an Add In with Visual Basic 6.0 (Page 1739)
Linking Add Ins (Page 1734)
3.5.4.2 Example: Creating an Add In with Visual Basic 6.0

Introduction

The program code in this example produces a file called "MyAddIn.DLL". So that the add-in will work in Graphics Designer, you must enable the "Load on Startup" checkbox for this add-in in the AddIn Manager of Graphics Designer. For this purpose you can also use the "LoadOnStartup" function in the add-in.

When you open Graphics Designer the add-in generates a user-defined menu. You can also use the "Tools" > "AddIns" menu to call the function contained in the add-in.

Requirement

In order to create an executable add-in from the sample code, "MS Visual Studio 6.0" must be installed on your computer. You must also have referenced "WinCC Graphics Designer 1.0 Type Library" in "MS Visual Studio 6.0":

![References - Project1](image)

Procedure

1. Open "MS Visual Studio 6.0" and create a new project. To create a project, go to the "New Project" dialog, select the "AddIn" entry and click on OK.
2. In Project Explorer, open the "Designer" folder and double click the entry called "Connect". The "Connect (AddIn Designer)" dialog opens.
4. In Project Explorer, open the "Designer" folder and use the shortcut menu to select the command "Display Code" for the "Connect" entry.
5. Replace the entire program code with the following program code:

```vba
Option Explicit
'-----------------------------------------------------
'Member Variables
'-----------------------------------------------------
'Reference to the add-in connection
Dim WithEvents ThisAddin As grafexe.AddInHook
'Reference to the Graphics Designer Application
Dim WithEvents GrafApp As grafexe.Application
'-----------------------------------------------------
'WithEvents AddInInstance IDTExtensibility2 (automatic)
'-----------------------------------------------------
'This method connects the add-in to the Graphic Designer Application
'-----------------------------------------------------
Private Sub AddInInstance_OnConnection(ByVal Application As Object, _
ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
ByVal AddInInst As Object, custom() As Variant)
On Error GoTo AddInInstance_OnConnection_Error
'--------------------------------------------------------
' Hook up to the Graphics Designer application.IAddInHookEvents interface.
' It is necessary referencing the application this add-in hooks up to
'--------------------------------------------------------
Dim GDApplication As grafexe.Application
Set GDApplication = Application
If (Not GDApplication Is Nothing) Then
'------------------------------------------------------
' Explanation on filters ( first parameter to AddIns.Attach() )
' sbAddInFilterExecute : Add-in is not shown in the AddIn menu
' (ThisAddin_Execute() is NOT called)
' sbAddInFilterNone : Add-in is shown in the AddIn menu
' and by clicking on the AddIn's menu entry
' ThisAddin_Execute() is called (see the figure below)
'------------------------------------------------------
Set ThisAddin = GDApplication.AddIns.Attach(sbAddInFilterNone, "Create Rectangle")
Set GrafApp = GDApplication
RegisterApplicationMenus
End If
```

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
1741
Exit Sub

AddInInstance_OnConnection_Error:
   MsgBox Err.Description
End Sub

'------------------------------------------------------------
'This method removes the add-in from VB by event disconnect
'------------------------------------------------------------
Private Sub AddInInstance_OnDisconnection(ByVal RemoveMode As AddInDesignerObjects.ext_DisconnectMode, _
custom() As Variant)
  On Error GoTo AddInInstance_OnDisconnection_Error
  If (RemoveMode = ext_dm_UserClosed) Then
     RemoveApplicationMenus
  End If

  ' Release reference to IAddInHookEvents interface - Important
  '------------------------------------------------------------
  Set ThisAddin = Nothing
  Set GrafApp = Nothing
  Exit Sub

AddInInstance_OnDisconnection_Error:
   MsgBox Err.Description
End Sub

'----------------------------------------------------------------
'This method describes the 2nd way to make add-in functions
available in Graphics Designer
'----------------------------------------------------------------
Private Sub RegisterApplicationMenus()
   Dim objDocMenu As HMIMenu
   Dim objMenuItem As HMIMenuItem
   Set objDocMenu = GrafApp.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
   Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "My first menu entry")
   Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second menu entry")

   Set objMenuItem = Nothing
   Set objDocMenu = Nothing
End Sub

'----------------------------------------------------------------
'This method removes the AddIn menus available in Graphics Designer
'----------------------------------------------------------------
Private Sub RemoveApplicationMenus()
    Dim objDocMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem

    For Each objMenuItem In GrafApp.CustomMenus("DocMenu1").MenuItems
        Set objMenuItem = Nothing
    Next objMenuItem

    GrafApp.CustomMenus("DocMenu1").Delete
    Set objMenuItem = Nothing
    Set objDocMenu = Nothing
End Sub

Private Sub AddInInstance_Terminate()
    ' --------------------------------------------------
    ' Release reference to IAddInHookEvents interface - Important
    ' --------------------------------------------------
    Set ThisAddin = Nothing
    Set GrafApp = Nothing
End Sub

Private Sub GrafApp_MenuItemClicked(ByVal MenuItem As grafexe.IHMIMenuItem)
    Select Case MenuItem.Key
        Case "dmItem1_1"
            TestCall1
        Case "dmItem1_2"
            TestCall2
        Case Else
            Debug.Assert False
    End Select
End Sub

'----------------------------------------------------------
'You can call both of the following procedures by clicking the menu command in the "DocMenu1"
'----------------------------------------------------------
Sub TestCall1()
    Call MsgBox("AddIn Menu: dmItem1_1 Clicked", vbInformation, "GrafApp_MenuItemClicked")
End Sub

Sub TestCall2()
    Call MsgBox("AddIn Menu: dmItem1_2 Clicked", vbInformation, "GrafApp_MenuItemClicked")
End Sub

'--------------------------------------------------------
'Registering an AddInHook creates an object which event can be executed by clicking "Extras\Macro\AddIns\<Registered Name>"
'--------------------------------------------------------
Private Sub ThisAddin_Execute()
    MsgBox("AddIn : Execute! Will create a new Rectangle now")
Dim NewShape As HMIObject
Set NewShape = GrafApp.ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With NewShape
  .Top = 40
  .Left = 40
  .BackColor = 255
End With
MsgBox (NewShape ObjectName)
End Sub

6. Create the add-in, and load it in the Graphics Designer.

Result

The next time you open it, Graphics Designer contains a user-defined menu titled "Doc_Menu_1". The menu "Tools > AddIns" contains an entry called "Create Rectangle", which pastes a rectangle into the active picture:

See also

How to Configure an AddIn in the Graphics Designer (Page 1737)
Example: Creating Add Ins (Page 1738)
3.6 VBA Reference

3.6.1 The object model of the Graphics Designer

3.6.1.1 VBA Reference

VBA Object Model

When you click an object name, you are shown a detailed description.

Note

The prefix "HMI" will be omitted from the following descriptions. Note that in the code you must prefix objects with "HMI", e.g. "HMSymbolLibrary".
See also

Events Object (Listing) (Page 1967)
SymbolLibraries Object (Listing) (Page 2073)
Actions Object (Listing) (Page 1898)
Application Object (Page 1909)
DataLanguage Object (Page 1941)
DataLanguages Object (Listing) (Page 1942)
Document Object (Page 1949)
Documents Object (Listing) (Page 1952)
Event Object (Page 1966)
HMIDefaultObjects Object (Listing) (Page 1985)
HMIObject Object (Page 1988)
HMIOBjects Object (Listing) (Page 1990)
FolderItem Object (Page 1972)
FolderItems Object (Listing) (Page 1973)
VBA Reference: ActionDynamic (Page 1747)
VBA Reference: HMIOBjects (Page 1749)
VBA Reference: Languages (Page 1751)
Layer Object (Page 2001)
Layers Object (Listing) (Page 2003)
Menu Object (Page 2009)
Menus Object (Listing) (Page 2011)
MenuItem Object (Page 2013)
MenuItems Object (Listing) (Page 2015)
Properties Object (Listing) (Page 2041)
Toolbar Object (Page 2080)
Toolbars Object (Listing) (Page 2081)
ToolbarItem Object (Page 2083)
ToolbarItems Object (Listing) (Page 2085)
View Object (Page 2102)
Views Object (Listing) (Page 2103)
SelectedObjects object (Listing) (Page 2060)
SymbolLibrary Object (Page 2074)
Property Object (Page 2042)
3.6.1.2 VBA Reference: ActionDynamic

VBA Object Model: ActionDynamic

"ActionDynamic" represents the interface port for dynamics and actions such as scripts, the dynamic dialog, the direct connection and the triggers.

When you click an object name, you are shown a detailed description.
BitResultInfo Object (Page 1919)
Actions Object (Listing) (Page 1898)
DestLink Object (Page 1945)
DirectConnection Object (Page 1947)
DynamicDialog Object (Page 1955)
QualityCodeStateValue Object (Page 2044)
QualityCodeStateValues Object (Listing) (Page 2046)
ScriptInfo Object (Page 2058)
SourceLink Object (Page 2065)
Trigger Object (Page 2087)
VariableStateValue Object (Page 2096)
VariableStateValues Object (Listing) (Page 2097)
VariableTrigger Object (Page 2099)
VariableTriggers Object (Listing) (Page 2101)
ActionType property (Page 2107)
DynamicStateType property (Page 2219)
3.6.1.3 VBA Reference: HMIObjects

VBA Object Model: HMIObjects

When you click an object name, you are shown a detailed description.

Object and List
Object

- Detail view available.
- Multilingual object configuration is possible.

* Not in DefaultObjects list.
See also

VBA Reference (Page 1745)
PolyLine Object (Page 2038)
GroupDisplay Object (Page 1980)
3DBarGraph Object (Page 1894)
ActiveXControl Object (Page 1900)
ApplicationWindow Object (Page 1911)
Button Object (Page 1921)
CheckBox Object (Page 1924)
Circle Object (Page 1928)
CircularArc Object (Page 1931)
Line Object (Page 2004)
OLEObject Object (Page 2023)
OptionGroup Object (Page 2025)
PictureWindow Object (Page 2029)
PieSegment Object (Page 2032)
Polygon Object (Page 2035)
Property Object (Page 2042)
Rectangle Object (Page 2049)
RoundButton Object (Page 2052)
RoundRectangle Object (Page 2055)
Slider object (Page 2062)
StaticText Object (Page 2067)
StatusDisplay Object (Page 2070)
TextList Object (Page 2076)
Ellipse Object (Page 1957)
EllipseArc Object (Page 1960)
EllipseSegment Object (Page 1963)
GraphicObject Object (Page 1975)
Group Object (Page 1978)
HMIDefaultObjects Object (Listing) (Page 1985)
HMIOBJECT Object (Page 1988)
HMIOBJECTs Object (Listing) (Page 1990)
IOField Object (Page 1992)
BarGraph Object (Page 1914)
3.6.1.4 VBA Reference: Languages

VBA Object Model: Languages

When you click an object name, you are shown a detailed description.

See also

VBA Reference (Page 1745)
LanguageFont Object (Page 1996)
LanguageFonts Object (Listing) (Page 1997)
LanguageText Object (Page 1999)
LanguageTexts Object (Listing) (Page 2000)
3.6.1.5 Events

A-D

Activated event

Description

Occurred when a picture is activated in the Graphics Designer. This happens when you switch between two pictures, for example.

Syntax

Document_Activated(CancelForwarding As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output when the picture is activated:

```
Private Sub Document_Activated(CancelForwarding As Boolean)
'VBA76
MsgBox "The document got the focus." & vbCrLf &_
"This event (Document_Activated) is raised by the document itself"
End Sub
```

See also

VBA Reference (Page 1745)
Event Handling (Page 1709)

BeforeClose Events

Description

Occurred immediately before a picture is closed.
syntax

Document_BeforeClose(Cancel As Boolean, CancelForwarding As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded.</td>
</tr>
<tr>
<td></td>
<td>Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output before the picture is closed:

Private Sub Document_BeforeClose(Cancel As Boolean, CancelForwarding As Boolean)
    'VBA77
    MsgBox "Event Document_BeforeClose is raised"
End Sub

See also

VBA Reference (Page 1745)

BeforeDocumentClose Event

Description

Occurs immediately before the picture is closed.

syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_BeforeDocumentClose(Document As HMIDocument, Cancel As Boolean)
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMIDocument)</td>
<td>The picture that is going to be closed.</td>
</tr>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub

In the following example a message is output before the picture is closed:

Private Sub objGDApplication_BeforeDocumentClose(ByVal Document As IHMIDocument, Cancel As Boolean)
    'VBA78
    MsgBox "The document " & Document.Name & " will be closed after press ok"
End Sub
```

See also

VBA Reference (Page 1745)

BeforeDocumentSave event

Description

Occurs immediately before the picture is saved.

Syntax

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
</table>
To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```vba
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for <Name>.
objGDApplication_BeforeDocumentSave (Document As HMIDocument, Cancel As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMIDocument)</td>
<td>The picture that is going to be closed.</td>
</tr>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output before the picture is closed:

```
Private Sub objGDApplication_BeforeDocumentSave(ByVal Document As IHMIDocument, Cancel As Boolean)
    'VBA79
    MsgBox Document.Name & "-saving will start after press ok."
End Sub
```

See also

VBA Reference (Page 1745)

BeforeHMIObjectDelete-Ereignis

Description

Occurs immediately before an object in a picture is deleted.

syntax

```
BeforeHMIObjectDelete(ByVal HMIObject As IHMIObject, Cancel As Boolean, CancelForwarding As Boolean)
```
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIObject (IHMIObject)</td>
<td>Identifies the object to be deleted.</td>
</tr>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output identifying the object to be deleted:

```vba
Private Sub Document_BeforeHMIObjectDelete(ByVal HMIObject As IHMIObject, Cancel As Boolean, CancelForwarding As Boolean)
    'VBA80
    Dim strObjName As String
    Dim strAnswer As String
    ' 
    ' "strObjName" contains the name of the deleted object
    strObjName = HMIObject.ObjectName
    strAnswer = MsgBox("Are you sure to delete " & strObjName & "?", vbYesNo)
    If strAnswer = vbNo Then
        'if pressed "No" -> set Cancel to true for prevent delete
        Cancel = True
    End If
    End Sub
```

See also

VBA Reference (Page 1745)

BeforeLibraryFolderDelete event

Description

Occurs immediately before a folder in the components library is deleted.

syntax

```
Note
```

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```vba
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDAutomation" is substituted for <Name>.
objGDApplication_BeforeLibraryFolderDelete(LibObject As HMIFolderItem, Cancel As Boolean)

Parameter (Optional)

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The folder that is going to be deleted.</td>
</tr>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output before a folder in the components library is deleted:

```vba
Private Sub objGDApplication_BeforeLibraryFolderDelete(ByVal LibObject As HMIFolderItem, Cancel As Boolean)
'VBA81
MsgBox "The library-folder " & LibObject.Name & " will be delete..."
End Sub
```

See also

VBA Reference (Page 1745)

BeforeLibraryObjectDelete event

Description

Occurs immediately before an object in the components library is deleted.
To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for `<Name>`.

```
objGDApplication_BeforeLibraryObjectDelete(LibObject As HMIFolderItem, Cancel As Boolean)
```

### Parameter (Optional)

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The object that is going to be deleted.</td>
</tr>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>

### Example:

Carry out the following procedure so that the example shown below will work:

```
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output before a folder in the components library is deleted:

```
Private Sub objGDApplication_BeforeLibraryObjectDelete(ByVal LibObject As HMIFolderItem, Cancel As Boolean)
    'VBA82
    MsgBox "The object " & LibObject.Name & " will be delete..."
End Sub
```

### See also

VBA Reference (Page 1745)
BeforeQuit Event

Description

Occurs immediately before the Graphics Designer is closed.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_BeforeQuit(Cancel As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub

In this example a message is output shortly before the Graphics Designer is closed.

Private Sub objGDApplication_BeforeQuit(Cancel As Boolean)
' VBA83
MsgBox "The Graphics Designer will be shut down"
End Sub

See also

VBA Reference (Page 1745)
BeforeSave Event

Description

Occurs immediately before a picture is saved.

Syntax

Document_BeforeSave(Cancel As Boolean, CancelForwarding As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output before the picture is saved:

Private Sub Document_BeforeSave(Cancel As Boolean, CancelForwarding As Boolean)
    'VBA84
    MsgBox "The document will be saved..."
End Sub

See also

VBA Reference (Page 1745)

BeforeVisibleFalse event

Description

Occurs immediately before the Graphics Designer application is set from Visible to Invisible.

Syntax

Document_BeforeVisibleFalse(Cancel As Boolean, CancelForwarding As Boolean)
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>
| CancelForwarding (Boolean)           | TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

```
--
```

See also

VBA Reference (Page 1745)

ConnectionEvent Event

Description

Occurs when two objects are connected via the connector.

syntax

```
ConnectionEvent(eConnEventType, HMIConnector, HMIClusteredObject, CancelProcess, CancelForwarding)
```

Parameter (Optional)

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eConnEventType (HMIConnectionEventType)</td>
<td>--</td>
</tr>
<tr>
<td>HMIConnector (HMIObject)</td>
<td>--</td>
</tr>
<tr>
<td>HMIClusteredObject (HMIObject)</td>
<td>--</td>
</tr>
<tr>
<td>CancelProcess (Boolean)</td>
<td>TRUE if command processing is to be canceled.</td>
</tr>
</tbody>
</table>
| CancelForwarding (Boolean)             | TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

```
--
```

See also

VBA Reference (Page 1745)
DataLanguageChanged Event

Description

Occurs when the project language has been changed.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for <Name>.

```
objGDApplication_DataLanguageChanged(lCID As Long)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lCID (Long)</td>
<td>The project language identifier</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub
```

In the following example the newly set project language is output:

```
Private Sub objGDApplication_DataLanguageChanged(ByVal lCID As Long)
    'VBA87
    MsgBox "The datalanguage is changed to " & Application.CurrentDataLanguage & "."
End Sub
```

See also

- Language-Dependent Configuration with VBA (Page 1624)
- VBA Reference (Page 1745)
DesktopLanguageChanged event

Description

Occurs when the user interface language has been changed.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_DesktopLanguageChanged(lCID As Long)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lCID (Long)</td>
<td>The user interface language identifier</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub

In the following example the newly set desktop language is output:

Private Sub objGDApplication_DesktopLanguageChanged(ByVal lCID As Long)
'VBA88
MsgBox "The desktop-language is changed to " & Application.CurrentDesktopLanguage & "."
End Sub

See also

VBA Reference (Page 1745)
Language-Dependent Configuration with VBA (Page 1624)
DocumentActivated Event

Description

Occurs when a picture is activated in the Graphics Designer. This happens when you switch between two pictures, for example.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for <Name>.

```
objGDApplication_DocumentActivated(Document As HMIDocument)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMIDocument)</td>
<td>The picture that is to be activated.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```
Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output identifying the picture that has been activated:

```
Private Sub objGDApplication_DocumentActivated(ByVal Document As IHMIDocument)
'VBA89
MsgBox "The document " & Document.Name & " got the focus." & vbCrLf & _
"This event is raised by the application."
End Sub
```

See also

VBA Reference (Page 1745)
**DocumentCreated Event**

**Description**

Occurs when a new picture has been created in the Graphics Designer.

**Syntax**

```
Note
To ensure that the application-specific event is available in the project, the application must
be made known to Graphics Designer. This is done by means of the following statement:
Dim WithEvents <Name> As grafexe.Application
In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_DocumentCreated(Document As HMIDocument)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMIDocument)</td>
<td>The picture that has been created.</td>
</tr>
</tbody>
</table>

**Example:**

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub
```

In the following example the name of the newly created picture is output:

```vba
Private Sub objGDApplication_DocumentCreated(ByVal Document As IHMIDocument)
    'VBA90
    MsgBox Document.Name & " will be created."
End Sub
```

**See also**

VBA Reference (Page 1745)
DocumentOpened Event

Description

Occurs when a picture has been opened.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_DocumentOpened(Document As HMIDocument)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMIDocument)</td>
<td>The picture that has been opened.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub

In the following example a message is output identifying the picture that has been opened:

Private Sub objGDApplication_DocumentOpened(ByVal Document As IHMIDocument)
    'VBA91
    MsgBox Document.Name & " is opened."
End Sub

See also

VBA Reference (Page 1745)
DocumentSaved Event

Description

Occurs when a picture has been saved in the Graphics Designer.

syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for <Name>.

```
objGDApplication_DocumentSaved(Document As HMI.Document)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (HMI.Document)</td>
<td>The picture that has been saved.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output identifying the picture that has been saved:

```vba
Private Sub objGDApplication_DocumentSaved(ByVal Document As IHMI.Document)
' VBA92
MsgBox Document.Name & " is saved."
End Sub
```

See also

VBA Reference (Page 1745)
DocumentPropertyChanged event

Description

Occurs when a picture property is changed.

Syntax

```vba
Document_DocumentPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (IHMIProperty)</td>
<td>Identifies the changed property.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded.</td>
</tr>
<tr>
<td></td>
<td>Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output identifying the picture property being changed:

```vba
Private Sub Document_DocumentPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
'VBA93
Dim strPropName As String
"strPropName" contains the name of the modified property
strPropName = Property.Name
MsgBox "The picture-property " & strPropName & " is modified..."
End Sub
```

See also

VBA Reference (Page 1745)

F-Z

HMIObjectAdded Event

Description

Occurs when an object is added.
syntax

Document_HMIObjectAdded(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIObject (IHMIObject)</td>
<td>Identifies the object being added.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output identifying the object that has been added:

Private Sub Document_HMIObjectAdded(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)
    'VBA94
    Dim strObjName As String
    
    '"strObjName" contains the name of the added object
    strObjName = HMIObject.ObjectName
    MsgBox "Object " & strObjName & " is added..."
    End Sub

See also

VBA Reference (Page 1745)

HMIObjectMoved Event

Description

Occurs when an object is moved.

syntax

Document_HMIObjectMoved(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIObject (IHMIObject)</td>
<td>Identifies the object being moved.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output identifying the object that has been moved:

Private Sub Document_HMIObjectMoved(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)
'VBA95
Dim strObjName As String
'
'"strObjName" contains the name of the moved object
strObjName = HMIObject.ObjectName
MsgBox "Object " & strObjName & " was moved..."
End Sub

See also

VBA Reference (Page 1745)

HMIObjectPropertyChanged Event

Description

Occurs when an object property is changed.

Syntax

Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (IHMIProperty)</td>
<td>Identifies the changed property.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>
Example:

In the following example a message is output identifying the object property that has been changed:

```vba
Private Sub Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
    'VBA96
    Dim strObjProp As String
    Dim strObjName As String
    Dim varPropValue As Variant
    
    '"strObjProp" contains the name of the modified property
    '"varPropValue" contains the new value
    strObjProp = Property.Name
    varPropValue = Property.value
    
    '"strObjName" contains the name of the selected object,
    'which property is modified
    strObjName = Property.Application.ActiveDocument.Selection(1).ObjectName
    MsgBox "The property " & strObjProp & " of object " & strObjName & " is modified... " & vbCrLf & "The new value is: " & varPropValue
End Sub
```

See also

VBA Reference (Page 1745)

HMIObjectResized Event

Description

Occurs when the size of an object is changed.

Syntax

```vba
Document_HMIObjectResized(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIObject (IHMIObject)</td>
<td>Identifies the object that is being resized.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>
Example:

In the following example a message is output when an object has been resized:

Private Sub Document_HMIObjectResized(ByVal HMIObject As IHMIObject, CancelForwarding As Boolean)
  'VBA97
  Dim strObjName As String
  ' "strObjName" contains the name of the modified object
  strObjName = HMIObject.ObjectName
  MsgBox "The size of " & strObjName & " was modified..."
End Sub

See also

VBA Reference (Page 1745)

LibraryFolderRenamed Event

Description

Occurs when a folder in the components library has been renamed.

Syntax

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_LibraryFolderRenamed(LibObject As HMIFolderItem, OldName As String)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The renamed folder.</td>
</tr>
<tr>
<td>OldName (String)</td>
<td>The original name of the renamed folder.</td>
</tr>
</tbody>
</table>
Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub

In the following example the old and new folder names are output:

Private Sub objGDApplication_LibraryFolderRenamed(ByVal LibObject As HMIFolderItem, ByVal OldName As String)
' VBA98
MsgBox "The Library-folder " & OldName & " is renamed in: " & LibObject.DisplayName
End Sub

See also

VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)

LibraryObjectRenamed Event

Description

Occurs when an object in the components library has been renamed.

Syntax

Dim WithEvents <Name> As grafexe.Application
In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_LibraryObjectRenamed(LibObject As HMIFolderItem, OldName As String)
### Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The renamed object.</td>
</tr>
<tr>
<td>OldName (String)</td>
<td>The original name of the renamed object.</td>
</tr>
</tbody>
</table>

### Example:

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub
```

In the following example the old and new object names are output:

```vba
Private Sub objGDApplication_LibraryObjectRenamed(ByVal LibObject As IHMIFolderItem, ByVal OldName As String)
    'VBA99
    MsgBox "The object " & OldName & " is renamed in: " & LibObject.DisplayName
End Sub
```

### See also

- VBA Reference (Page 1745)
- Accessing the component library with VBA (Page 1646)

### LibraryObjectAdded Event

#### Description

Occurs when an object has been added to the components library.

#### Syntax

```vba
HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
```
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (IHMIFileItem)</td>
<td>Identifies the library object.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output when an object has been added to the components library:

Private Sub Document_LibraryObjectAdded(ByVal LibObject As IHMIFileItem, CancelForwarding As Boolean)
'VBA100
Dim strObjName As String
'"strObjName" contains the name of the added object
strObjName = LibObject.DisplayName
MsgBox "Object " & strObjName & " was added to the picture."
End Sub

See also

VBA Reference (Page 1745)

MenuItemClicked Event

Description

Occurs when an entry in a user-defined menu is clicked.

Note

This event is both application-specific and document-specific.

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

syntax

Document_MenuItemClicked(ByVal MenuItem As IHMIFileItem)
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MenuItem (IHMIMenuItem)</td>
<td>Identifies the user-defined menu.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

```vba
Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
```

In the following example a message is output when the first entry in a user-defined menu is clicked:

```vba
Private Sub Document_MenuItemClicked(ByVal MenuItem As IHMIMenuItem)
' VBA101
Dim objMenuItem As HMIMenuItem
Dim varMenuItemKey As Variant
Set objMenuItem = MenuItem

"objMenuItem" contains the clicked menu-item
"varMenuItemKey" contains the value of parameter "Key"
' from the clicked user-defined menu-item
varMenuItemKey = objMenuItem.Key
Select Case MenuItem.Key
Case "mItem1_1"
MsgBox "The first menu-item was clicked!"
End Select
End Sub
```

See also

- How to assign VBA macros to menus and toolbars (Page 1642)
- VBA Reference (Page 1745)

NewLibraryFolder Event

Description

Occurs when a folder has been created in the components library.
syntax

Note
To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

objGDApplication_NewLibraryFolder(LibObject As HMIFolderItem)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The newly created folder.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub

In the following example the new folder name is output:

Private Sub objGDApplication_NewLibraryFolder(ByVal LibObject As IHMIFolderItem)
    'VBA102
    MsgBox "The library-folder " & LibObject.DisplayName & " was added."
End Sub

See also

VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)

NewLibraryObject Event

Description

Occurs when an object has been created in the components library.
To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```
Dim WithEvents <Name> As grafexe.Application
```

In the following example, the name "objGDApplication" is substituted for <Name>.

```
objGDApplication_NewLibraryObject(LibObject As HMIFolderItem)
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibObject (HMIFolderItem)</td>
<td>The newly created object.</td>
</tr>
</tbody>
</table>

### Example:

Carry out the following procedure so that the example shown below will work:

```
Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
```

In the following example the new object name is output:

```
Private Sub objGDApplication_NewLibraryObject(ByVal LibObject As IHMIFolderItem)
'VBA103
MsgBox "The object " & LibObject.DisplayName & " was added."
End Sub
```

### See also

- VBA Reference (Page 1745)
- Accessing the component library with VBA (Page 1646)

### Opened Event

**Description**

Occurs when a picture is opened.
**syntax**

```
Document_Opened(CancelForwarding As Boolean)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

**Example:**

In the following example a message is output when the picture is opened:

```vba
Private Sub Document_Opened(CancelForwarding As Boolean)
    'VBA104
    MsgBox "The Document is open now..."
End Sub
```

**See also**

VBA Reference (Page 1745)

**Saved Event**

**Description**

Occurs after a picture has been saved.

**syntax**

```
Document_Saved(CancelForwarding As Boolean)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>
Example:

In the following example a message is output when the picture has been saved:

```vba
Private Sub Document_Saved(CancelForwarding As Boolean)
    'VBA105
    MsgBox "The document is saved..."
End Sub
```

See also

VBA Reference (Page 1745)

SelectionChanged Event

Description

Occurs when the selection has been changed.

Syntax

```
Document_SelectionChanged(CancelForwarding As Boolean)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded.</td>
</tr>
<tr>
<td></td>
<td>Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

In the following example a message is output when a new object has been selected:

```vba
Private Sub Document_SelectionChanged(CancelForwarding As Boolean)
    'VBA106
    MsgBox "The selection is changed..."
End Sub
```

See also

VBA Reference (Page 1745)
Started Event

Description

Occurs when the Graphics Designer has been started.

Syntax

objGDApplication_Started()

Note

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

Parameters

--

Example

Declare application.

Dim WithEvents objGDApplication As grafexe.Application

Set event tag.

Private Sub Document_Opened(CancelForwarding As Boolean)
    Set objGDApplication = Me.Application
End Sub

Query "Started" event and output message.

Private Sub objGDApplication_Started()
    'VBA107
    'This event is raised before objGDApplication_Started()
    MsgBox "The Graphics Designer is started!"
End Sub
See also

VBA Reference (Page 1745)

ToolBarItemClicked Event

Description

Occurs when an icon in a user-defined toolbar has been clicked

Note

This event is both application-specific and document-specific.

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

Dim WithEvents <Name> As grafexe.Application

Syntax

Document_ToolbarItemClicked(ByVal ToolbarItem As IHMIToolbarItem)

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolbarItem (IHMIToolbarItem)</td>
<td>Identifies the symbol.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
' This procedure have to execute with "F5" first
Set objGDApplication = grafexe.Application
End Sub
In the following example a message is output when the first user-defined icon is clicked:

```vba
Private Sub Document_ToolbarItemClicked(ByVal ToolbarItem As IHMIToolbarItem)
    'VBA108
    Dim objToolbarItem As HMIToolbarItem
    Dim varToolbarItemKey As Variant
    Set objToolbarItem = ToolbarItem
    ' "varToolbarItemKey" contains the value of parameter "Key"
    ' from the clicked userdefined toolbar-item
    varToolbarItemKey = objToolbarItem.Key
    Select Case varToolbarItemKey
        Case "tItem1_1"
            MsgBox "The first Toolbar-Icon was clicked!"
    End Select
End Sub
```

See also

- How to assign VBA macros to menus and toolbars (Page 1642)
- VBA Reference (Page 1745)

**ViewCreated Event**

**Description**

Occurs when a copy of a picture has been created.

**Note**

This event is both application-specific and document-specific.

To ensure that the application-specific event is available in the project, the application must be made known to Graphics Designer. This is done by means of the following statement:

```vba
Dim WithEvents <Name> As grafexe.Application
```

**syntax**

```vba
Document_ViewCreated(ByVal pView As IHMIView, CancelForwarding As Boolean)
```
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pView (IHMIView)</td>
<td>Identifies the copy of the picture.</td>
</tr>
<tr>
<td>CancelForwarding (Boolean)</td>
<td>TRUE if the event is not intended to be forwarded. Default setting is &quot;False&quot;.</td>
</tr>
</tbody>
</table>

Example:

Carry out the following procedure so that the example shown below will work:

Private Sub SetApplication()
    'This procedure have to execute with "F5" first
    Set objGDApplication = grafexe.Application
End Sub

In the following example the number of copy pictures is output when a new copy of the picture has been created.

Private Sub Document_ViewCreated(ByVal pView As IHMIView, CancelForwarding As Boolean)
    'VBA109
    Dim iViewCount As Integer
    'To read out the number of views
    MsgBox "A new copy of the picture (number " & iViewCount & ") was created."
End Sub

See also

VBA Reference (Page 1745)

WindowStateChange Event

Description

Occurs when the window size is changed (e.g. from "Minimized" to "Maximized").

syntax

objGDApplication_WindowStateChanged()
Parameter (Optional)

Example:
In the following example a message is output when the window size is changed:

Private Sub objGDApplication_WindowStateChanged()
    'VBA110
    MsgBox "The state of the application-window is changed!"
End Sub

See also
VBA Reference (Page 1745)

3.6.1.6 Methods

A-C

Activate Method

Description
Activates the specified object.

Syntax

Expression.Activate()

Expression
Necessary. An expression or element which returns an object of the "Application" or "View" type.

Parameters
--
Example:

In the following example a copy of the active picture is created and then activated:

```vba
Sub CreateAndActivateView()
    'VBA111
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
End Sub
```

See also

- View Object (Page 2102)
- Application Object (Page 1909)
- VBA Reference (Page 1745)

Add Method

Description

Adds another element to a listing.

The following table shows you the listings to which the Add method can be applied. The parameters and syntax for the respective Add methods can be found under “Methods”.

<table>
<thead>
<tr>
<th>Listing</th>
<th>Application for the Add Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalogResultInfos Listing</td>
<td>Adds a new, analog value range in the Dynamic dialog.</td>
</tr>
<tr>
<td>Documents Listing</td>
<td>Creates a new picture in the Graphics Designer</td>
</tr>
<tr>
<td>GroupedObjects Listing</td>
<td>Adds a new object to a group object.</td>
</tr>
<tr>
<td>Toolbars Listing</td>
<td>Creates a new, user-defined toolbar.</td>
</tr>
<tr>
<td>Tag Triggers Listing</td>
<td>Creates a new tag trigger.</td>
</tr>
<tr>
<td>Views Listing</td>
<td>Creates a copy of the specified picture.</td>
</tr>
</tbody>
</table>

See also

- Add Method (Views Listing) (Page 1793)
- Add Method (TagTriggers Listing) (Page 1792)
- Add Method (CustomToolbars Listing) (Page 1788)
- Add Method (GroupedObjects Listing) (Page 1791)
Add Method (AnalogResultInfos Listing)

**Description**

Adds a new, analog value range in the Dynamic dialog.

**Syntax**

*Expression*.Add(RangeTo, ResultValue)

**Expression**

Necessary. An expression or element which returns an object of the "AnalogResultInfos" type.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeTo (Variant)</td>
<td>The value range to which the change of property gives rise.</td>
</tr>
<tr>
<td>ResultValue (Variant)</td>
<td>The value to which the object property is assigned when the value range is reached.</td>
</tr>
</tbody>
</table>

**Example:**

In the following example the radius of a circle is given dynamics with the In the following example a tag name is assigned and three analog value ranges are created:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA112
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.Add 50, 40
        .AnalogResultInfos.Add 100, 80
        .AnalogResultInfos.ElseCase = 100
    End With
End Sub
```

The diagram shows the Dynamic dialog after the procedure has been carried out:
See also

DynamicDialog Object (Page 1955)
AnalogResultInfos Object (Listing) (Page 1908)
CreateDynamic Method (Page 1828)
How to dynamize a property with the Dynamic dialog (Page 1691)

Add Method (CustomToolbars Listing)

Description

Creates a new, user-defined toolbar. There is a difference between application-specific and picture-specific user-defined toolbars:

- Application-specific toolbar: This is linked to the Graphics Designer and is also only visible when all the pictures in the Graphics Designer are closed. "Place the VBA code in the document called "GlobalTemplateDocument" or "ProjectTemplateDocument" and use the Application property."
- Picture-specific toolbar: Is linked with a specific picture and remains visible as long as the picture is visible. Place the VBA code in the document called "ThisDocument" for the desired picture and use the ActiveDocument property.

Syntax

Expression.Add(Key)

Expression

Necessary. An expression or element which returns an object of the "CustomToolbars" type.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key (Variant)</td>
<td>Identifies the user-defined toolbar. Use unique names for &quot;Key&quot; (e.g. &quot;DocToolbar1&quot;)</td>
</tr>
</tbody>
</table>

Example:

In the following example a user-defined toolbar with two icons is created in the active picture. These icons are separated by a dividing line:

```vba
Sub AddDocumentSpecificCustomToolbar()
  'VBA115
  Dim objToolbar As HMIToolbar
  Dim objToolbarItem As HMIToolbarItem

  Set objToolbar = ActiveDocument.CustomToolbars.Add("DocToolbar")

  'Add toolbar-items to the userdefined toolbar
  Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "tItem1_1", "My first Symbol-Icon")
  Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(3, "tItem1_3", "My second Symbol-Icon")

  'Insert seperatorline between the two tollbaritems
  Set objToolbarItem = objToolbar.ToolbarItems.InsertSeparator(2, "tSeparator1_2")
  End Sub
```

See also

- Toolbars Object (Listing) (Page 2081)
- InsertToolbarItem Method (Page 1856)
- InsertSeparator Method (Page 1853)
- InsertFromMenuItem Method (Page 1848)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
Add Method (Documents Listing)

Description
Creates a new picture in the Graphics Designer

syntax

Expression.Add [HMIOpenDocumentType]

Expression
Necessary. An expression or element which returns an object of the "Documents" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIOpenDocumentType (HMIDocumentType)</td>
<td>Defines how the picture will be opened:</td>
</tr>
<tr>
<td></td>
<td>• HMIDocumentTypeVisible: Opens the picture for direct processing. This is the default setting if you do not specify the parameter.</td>
</tr>
<tr>
<td></td>
<td>• HMIDocumentTypeInvisible: Opens the picture in invisible mode, i.e. it is not displayed in the Graphics Designer. You can only address the picture via the Documents listing, and make it visible again by means of the Hide property.</td>
</tr>
</tbody>
</table>

Example:

In the following example a new picture is created in the Graphics Designer:

```vba
Sub AddNewDocument()
'VBA113
Application.Documents.Add hmiOpenDocumentTypeVisible
End Sub
```

See also

Hide Property (Page 2270)
Documents Object (Listing) (Page 1952)
VBA Reference (Page 1745)
Add Method (GroupedObjects Listing)

Description

Adds an existing object to the specified group object.

syntax

Expression/Add(Index)

Expression
Necessary. An expression or element which returns an object of the "GroupedObjects" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>The object that is intended to be added. You can either use the index number or the object name.</td>
</tr>
</tbody>
</table>
Example:

In this example the group object "My Group" is created from a number of objects. An ellipse segment is then added to the group object:

Sub CreateGroup()
  'VBA114
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Dim objEllipseSegment As HMIEllipseSegment
  Dim objGroup As HMIGroup

  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
  With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
  End With
  With objRectangle
    .Top = 80
    .Left = 80
    .Selected = True
  End With
  MsgBox "Objects selected!"
  Set objGroup = ActiveDocument.Selection.CreateGroup
  'Set name for new group-object
  'The name identifies the group-object
  objGroup.ObjectName = "My Group"
  'Add new object to active document...
  Set objEllipseSegment = ActiveDocument.HMIObjects.AddHMIObject("EllipseSegment", "HMIEllipseSegment")
  Set objGroup = ActiveDocument.HMIObjects("My Group")
  '...and add it to the group:
  objGroup.GroupedHMIObjects.Add ("EllipseSegment")
End Sub

See also

GroupedObjects Object (Listing) (Page 1983)

Add Method (TagTriggers Listing)

Description

  Creates a new tag trigger.
syntax

\[
Expression.Add(\text{VarName, Type})
\]

**Expression**

Necessary. An expression or element which returns an object of the "TagTriggers" type.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VarName (String)</td>
<td>The name of the tag that is intended to be used as a trigger. Please note that you have to create the tag in the Tag Selection dialog.</td>
</tr>
<tr>
<td>Type (CycleType)</td>
<td>This is the cycle type. Select the cycle type from a list in the VBA Editor when you use this method.</td>
</tr>
</tbody>
</table>

**Example:**

In the following example the radius of a circle is made dynamic using a trigger tag:

```vba
Sub DynamicWithVariableTriggerCycle()
    'VBA69
    Dim objVBScript As HMIScriptInfo
    Dim objVarTrigger As HMIVariableTrigger
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VariableTrigger", "HMICircle")
    Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    With objVBScript
        Set objVarTrigger = .Trigger.VariableTriggers.Add("VarTrigger", hmiVariableCycleType_10s)
        .SourceCode = ""
    End With
End Sub
```

**See also**

VariableTriggers Object (Listing) (Page 2101)

VBA Reference (Page 1745)

**Add Method (Views Listing)**

**Description**

Creates a copy of the specified picture.
syntax

Expression.Add()

Expression
Necessary. An expression or element which returns an object of the "Views" type.

Parameters

--

Example:

In the following example a copy of the active picture is created and then activated:

Sub CreateViewAndActivateView()
    'VBA117
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
End Sub

See also

Views Object (Listing) (Page 2103)
VBA Reference (Page 1745)

AddAction Method

Description

Configures an action on an object or property. This action is triggered when a defined event occurs.

syntax

Expression.Method(HMIActionCreationType)

Expression
Necessary. An expression or element which returns an object of the "Actions" type.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIActionCreationType (Variant)</td>
<td>Defines the action:</td>
</tr>
<tr>
<td></td>
<td>● hmiActionCreationTypeCScript: Configures a C action</td>
</tr>
<tr>
<td></td>
<td>● hmiActionCreationTypeVBScript: Configures a VBS action</td>
</tr>
<tr>
<td></td>
<td>● hmiActionCreationTypeDirectConnection: Configures a direct connection</td>
</tr>
</tbody>
</table>

Example:

In the following example a VBS action for changing the radius of a circle is configured:

```vba
Sub AddActionToPropertyTypeVBScript()
    'VBA118
    Dim objEvent As HMIEvent
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMICircle
    'Create circle in picture. By changing of property "Radius"
    'a VBS-action will be started:
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
    Set objEvent = objCircle.Radius.Events(1)
    Set objVBScript = objEvent.Actions.AddAction(hmiActionCreationTypeVBScript)
End Sub
```

See also

Event Object (Page 1966)
Actions Object (Listing) (Page 1898)

AddActiveXControl Method

Description

Adds a new ActiveXControl object to the "HMIObjects" listing. The object is inserted in the upper left corner of the specified picture.

Syntax

```vba
Expression.AddActiveXControl("ObjectName", "ProgID")
```

Expression

Required. An expression or element which returns an object of the "HMIObjects" type.
Parameter

<table>
<thead>
<tr>
<th>Parameter (data type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object. You can address the object by its name in a listing.</td>
</tr>
<tr>
<td>ProgID (String)</td>
<td>The ActiveX Control that is to be inserted.</td>
</tr>
</tbody>
</table>

Determining the ProgID

To determine the ProgID for an ActiveX control, go to the "Object Palette" in the Graphics Designer and in the Default tab under "Smart Objects" insert the control object into the picture. The "Insert a Control" dialog displays the path and ProgID for the selected control:

The following table shows a list of ProgIDs of WinCC controls that are installed by WinCC:

<table>
<thead>
<tr>
<th>Name of the WinCC control</th>
<th>ProgID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siemens HMI Symbol Library</td>
<td>SiemensHMI.SymbolLibrary.1</td>
</tr>
<tr>
<td>WinCC AlarmControl</td>
<td>CCAXAlarmControl.AxAlarmControl.1</td>
</tr>
<tr>
<td>WinCC digital/analog clock control</td>
<td>DACLOCK.DaclockCtrl.1</td>
</tr>
<tr>
<td>WinCC FunctionTrendControl</td>
<td>CCAXFunctionTrendControl.AxFUnctionTrendControl.1</td>
</tr>
<tr>
<td>WinCC gauge control</td>
<td>XGAUGE.XGaugeCtrl.1</td>
</tr>
<tr>
<td>WinCC media control</td>
<td>CCMediaControl.CCMediaControl.1</td>
</tr>
<tr>
<td>WinCC OnlineTableControl</td>
<td>CCAXOnlineTableControl.AxOnlineTableControl.1</td>
</tr>
<tr>
<td>WinCC OnlineTrendControl</td>
<td>CCAXOnlineTrendControl.AxOnlineTrendControl.1</td>
</tr>
<tr>
<td>WinCC push button control</td>
<td>PBUTTON.PbuttonCtrl.1</td>
</tr>
</tbody>
</table>
### Name of the WinCC control

<table>
<thead>
<tr>
<th>Name of the WinCC control</th>
<th>ProgID</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinCC slider control</td>
<td>SLIDER.SliderCtrl.1</td>
</tr>
<tr>
<td>WinCC RulerControl</td>
<td>CCAXTrendRulerControl.AxRulerControl.1</td>
</tr>
<tr>
<td>WinCC UserArchiveControl</td>
<td>CCAXUserArchiveControl.AxUserArchiveControl.1</td>
</tr>
</tbody>
</table>

### Example:

In the following example, the ActiveX Control "WinCC Gauge Control" is inserted in the active picture.

```vba
Sub AddActiveXControl()
    'VBA119
    Dim objActiveXControl As HMIActiveXControl
    Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge", "XGAUGE.XGaugeCtrl.1")
    With ActiveDocument
        .HMIObjects("WinCC_Gauge").Top = 40
        .HMIObjects("WinCC_Gauge").Left = 40
    End With
End Sub
```

**Note**

After executing the method, the Graphics Designer will not be fully shut down. The "Grafexe.exe" file remains in the memory. In order to restart the Graphics Designer, exit the "Grafexe.exe" application in the Task Manager.

### See also

- ActiveX controls (Page 1671)
- HMIObjects Object (Listing) (Page 1990)
- ActiveXControl Object (Page 1900)
- VBA Reference (Page 1745)

### AddDotNetControl method

**Description**

Adds a new ".Net-Control" object to the "HMIObjects" listing.

**Syntax**

```vba
Expression.AddDotNetControl(ObjectName, ControlType, InGAC, AssemblyInfo)
```
Expression

Necessary. An expression or element which returns an object of the "HMIObjects" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object. You can address the object by its name in a listing.</td>
</tr>
<tr>
<td>ControlType (String)</td>
<td>The namespace of the object.</td>
</tr>
<tr>
<td>InGAC (String)</td>
<td>TRUE: The object is registered in the Global Assembly Cache. FALSE: The object is not registered in the Global Assembly Cache.</td>
</tr>
<tr>
<td>AssemblyInfo (String)</td>
<td>If &quot;InGAC=TRUE&quot;, then the following information will be specified: Assembly, Version, Culture, PublicKeyToken. If &quot;InGAC=FALSE&quot;, only the path of the object is specified in &quot;Assembly&quot;.</td>
</tr>
</tbody>
</table>

Example

In the following example, the "NETControl" object from the Global Assembly Cache is inserted in the active picture.

'VBA851
Dim DotNetControl As HMIDotNetControl

AddFolder Method

Description

Creates a new folder in the components library. The FolderItem object of the "Folder" type is added to the FolderItems listing.

The new folder created in this way receives the internal name "FolderX", where "X" stands for a consecutive number, starting with 1. Use the internal name to address the folder in the FolderItems listing.

syntax

Expression.AddFolder(DefaultName)
Expression

Necessary. An expression or element which returns an object of the "FolderItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultName (String)</td>
<td>The name of the folder that is to be created.</td>
</tr>
</tbody>
</table>

Example:

In the following example the folder "My Folder" will be created in the "Project Library":

```vba
Sub AddNewFolderToProjectLibrary()
  'VBA120
  Dim objProjectLib As HMISymbolLibrary
  Set objProjectLib = Application.SymbolLibraries(2)
  objProjectLib.FolderItems.AddFolder("My Folder")
End Sub
```

See also

- SymbolLibrary Object (Page 2074)
- FolderItems Object (Listing) (Page 1973)
- VBA Reference (Page 1745)
- Accessing the component library with VBA (Page 1646)

AddFromClipboard Method

Description

Copies an object from the clipboard into a folder in the Components Library. The FolderItem object of the "Item" type is added to the FolderItems listing.

Note

The clipboard must contain objects from the Graphics Designer. Other contents (such as ASCII text) will not be pasted.

Syntax

```vba
Expression.AddFromClipboard(DefaultName)
```
Expression

Necessary. An expression or element which returns an object of the "FolderItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultName (String)</td>
<td>The name to be given to the object pasted into the components library.</td>
</tr>
</tbody>
</table>

Example:

In the following example the object "PC" from the "Global Library" will be copied into the folder "Folder 3" in the "Project Library":

```vba
Sub CopyObjectFromGlobalLibraryToProjectLibrary()
    'VBA121
    Dim objGlobalLib As HMISymbolLibrary
    Dim objProjectLib As HMISymbolLibrary
    Set objGlobalLib = Application.SymbolLibraries(1)
    Set objProjectLib = Application.SymbolLibraries(2)
    objProjectLib.FolderItems.AddFolder ("My Folder3")
    'copy object from "Global Library" to clipboard
    With objGlobalLib
        .FolderItems(2).Folder.Item(2).Folder.Item(1).CopyToClipboard
    End With
    'paste object from clipboard into "Project Library"
    objProjectLib.FolderItems(objProjectLib.FindByDisplayName("My Folder3").Name).Folder.AddFromClipBoard ("Copy of PC/PLC")
End Sub
```

See also

FolderItems Object (Listing) (Page 1973)
SymbolLibrary Object (Page 2074)
VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)
AddHMIObject Method

Description

Adds a new standard, smart or Windows object to the "HMIObjects" listing. The object is inserted in the upper left corner of the specified picture.

Note

Use the AddActiveXControl method to insert an ActiveXControl.
Use the AddOLEObject method to insert an OLE Element.

Syntax

Expression.AddHMIObject("ObjectName", "ProgID")

Expression
Necessary. An expression or element which returns an object of the "HMIObjects" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object. You can address the object by its name in a listing.</td>
</tr>
<tr>
<td>ProgID (String)</td>
<td>The object type that is to be inserted. &quot;Obtain the &quot;ProgID&quot; by prefixing the VBA object name with &quot;HMI&quot; (e.g. HMICircle or HMIRectangle)</td>
</tr>
</tbody>
</table>

Example:

In the following example a circle will be inserted into the active picture and its background color set to "Red":

```vba
Sub AddCircleToActiveDocument()
'VBA122
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("VBA_Circle", "HMICircle")
objCircle.BackColor = RGB(255, 0, 0)
End Sub
```

See also

PieSegment Object (Page 2032)
TextList Object (Page 2076)
AddItem Method

Description

Copies an object from the specified picture into a folder in the Components Library. The FolderItem object of the "Item" type is added to the FolderItems listing.

Syntax

Expression.Folder.AddItem "DefaultName", pHMIObject
Expression
Necessary. An expression or element which returns an object of the "FolderItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultName (String)</td>
<td>The name to be given to the object pasted into the components library.</td>
</tr>
<tr>
<td>pHMIOBJECT (HMIOBJECT)</td>
<td>The object that is to be inserted into the Components Library from the specified picture.</td>
</tr>
</tbody>
</table>

Example:

In the following example a circle will be copied into the "Project Library". For this purpose the circle will be pasted into the active picture and the folder "My Folder 2" will

```vba
Sub VBA123()
    'VBA123
    Dim objProjectLib As HMISymbolLibrary
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle", "HMICircle")
    Set objProjectLib = Application.SymbolLibraries(2)
    objProjectLib.FolderItems.AddFolder ("My Folder2")
    objProjectLib.FindByDisplayName("My Folder2").Folder.AddItem "ProjectLib Circle",
    ActiveDocument.HMIObjects("Circle")
End Sub
```

See also
- FolderItems Object (Listing) (Page 1973)
- SymbolLibrary Object (Page 2074)
- VBA Reference (Page 1745)

Accessing the component library with VBA (Page 1646)

AddOLEObject Method

Description
Adds a new OLE Element to the "HMIOBJECTs" listing. The object is inserted in the upper left corner of the specified picture.
syntax

Expression.AddOLEObject(ObjectName, ServerName, [CreationType], [UseSymbol])

Expression

Necessary. An expression or element which returns an object of the "HMIObjects" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object. You can address the object by its name in a listing.</td>
</tr>
<tr>
<td>ServerName (String)</td>
<td>The name of the application which is to contain the OLE Element, or the file name complete with path. The value for &quot;ServerName&quot; corresponds to the &quot;Object Type&quot; in the &quot;Insert Object&quot; dialog:</td>
</tr>
<tr>
<td>CreationType (HMIOLEObjectCreationType-)</td>
<td>Defines whether the OLE Element will be newly created or an existing file will be used:</td>
</tr>
<tr>
<td></td>
<td>• HMIOLEObjectCreationTypeDirect: Corresponds to setting &quot;Create New&quot;. This setting is used if you do not specify the parameter.</td>
</tr>
<tr>
<td></td>
<td>• HMIOLEObjectCreationTypeByLink: Corresponds to setting &quot;Create from File&quot;. This creates a copy of the file. Any changes made to the OLE Element have no effect on the original file. Assign a name to the file via the &quot;ServerName&quot; parameter.</td>
</tr>
<tr>
<td></td>
<td>• HMIOLEObjectCreationTypeByLinkWithReference: Same as above, except that changes in OLE Element affect the original file. Assign a name to the file via the &quot;ServerName&quot; parameter.</td>
</tr>
<tr>
<td>UseSymbol (Boolean)</td>
<td>TRUE if the standard icon for the file type is to be used. Double clicking on the icon then opens the associated application. The default setting for this parameter is FALSE.</td>
</tr>
</tbody>
</table>
Example:

In the following example, an OLE Element containing a Wordpad document will be inserted into the active picture:

Sub AddOLEObjectToActiveDocument()
'VBA124
Dim objOLEObject As HMIOLEObject
Set objOLEObject = ActiveDocument.HMIObjects.AddOLEObject("MS Wordpad Document", "Wordpad.Document.1")
End Sub

In the following example, the AddOLEObject method will be used and the "HMIOLEObjectCreationTypeByLink" parameter will be specified:

Sub AddOLEObjectByLink()
'VBA805
Dim objOLEObject As HMIOLEObject
Dim strFilename As String
'
'Add OLEObject by filename. In this case, the filename has to contain filename and path.
'Replace the definition of strFilename with a filename with path existing on your system
strFilename = Application.ApplicationDataPath & "Test.bmp"
Set objOLEObject = ActiveDocument.HMIObjects.AddOLEObject("OLEObject1", strFilename, hmiOLEObjectCreationTypeByLink, False)
End Sub

In the following example, the AddOLEObject method will be used and the "HMIOLEObjectCreationTypeByLinkWithReference" parameter will be specified:

Sub AddOLEObjectByLinkWithReference()
'VBA806
Dim objOLEObject As HMIOLEObject
Dim strFilename As String
'
'Add OLEObject by filename. In this case, the filename has to contain filename and path.
'Replace the definition of strFilename with a filename with path existing on your system
strFilename = Application.ApplicationDataPath & "Test.bmp"
Set objOLEObject = ActiveDocument.HMIObjects.AddOLEObject("OLEObject1", strFilename, hmiOLEObjectCreationTypeByLinkWithReference, True)
End Sub
AddWPFControl method

Description

Adds a new "WPF-Control" object to the "HMIObjects" listing.

Syntax

Expression.AddWPFControl(ObjectName, ControlType, InGAC, AssemblyInfo)

Expression

Necessary. An expression or element which returns an object of the "HMIObjects" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object. You can address the object by its name in a listing.</td>
</tr>
<tr>
<td>ControlType (String)</td>
<td>The namespace of the object.</td>
</tr>
<tr>
<td>InGAC (String)</td>
<td>TRUE: The object is registered in the Global Assembly Cache. FALSE: The object is not registered in the Global Assembly Cache.</td>
</tr>
<tr>
<td>AssemblyInfo (String)</td>
<td>If &quot;InGAC=TRUE&quot;, then the following information will be specified: Assembly Version Culture PublicKeyToken If &quot;InGAC=FALSE&quot;, only the path of the object is specified in &quot;Assembly&quot;.</td>
</tr>
</tbody>
</table>

Example

In the following example, the "WPF Control" object outside the Global Assembly Cache is inserted in the active picture.

'VBA852
Dim WPFControl As HMIWPFControl
Set WPFControl = ActiveDocument.HMIObjects.AddWPFControl("MyWPFVBAControl", "WinCCWPFControl.TestControl", False, "Assembly=Z:\TestControl\WinCCWPFControl.dll")
AlignBottom Method

Description

Aligns the objects selected in the specified picture with In so doing the alignment is oriented on the first object that you select.

Syntax

Expression.AlignBottom()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects are inserted at different positions in the current picture and then aligned with the bottom:

```vba
Sub AlignSelectedObjectsBottom()
    'VBA125
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIOObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIOObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    ActiveDocument.Selection.AlignBottom
End Sub
```

See also

SelectedObjects object (Listing) (Page 2060)

VBA Reference (Page 1745)
AlignLeft Method

Description

Left-justifies the objects selected in the specified picture. In so doing the alignment is oriented on the first object that you select.

Syntax

Expression.AlignLeft()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects are inserted at different positions in the current picture and then aligned to the left:

Sub AlignSelectedObjectsLeft()
    'VBA126
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    ActiveDocument.Selection.AlignLeft
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)
AlignRight Method

Description

Right-justifies the objects selected in the specified picture. In so doing the alignment is oriented on the first object that you select.

Syntax

Expression.AlignRight()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects are inserted at different positions in the current picture and then aligned to the right:

Sub AlignSelectedObjectsRight()
'VBA127
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.AlignRight
End Sub

See also

SelectedObjects object (Listing) (Page 2060)

VBA Reference (Page 1745)
AlignTop Method

Description

Aligns the objects selected in the specified picture with In so doing the alignment is oriented on the first object that you select.

Syntax

Expression.AlignTop()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects are inserted at different positions in the current picture and then aligned with the top:

Sub AlignSelectedObjectsTop()
    'VBA128
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    ActiveDocument.Selection.AlignTop
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)
ArrangeMinimizedWindows Method

Description

Arranges all minimized pictures on the lower margin of the Graphics Designer.

Syntax

Expression.ArrangeMinimizedWindows()

Expression

Necessary. An expression or element which returns an object of the "Application" type.

Parameters

--

Example:

In the following example all minimized pictures are arranged on the lower margin of the
Graphics Designer. For this example to work, you must have minimized a number of pictures
in the Graphics Designer:

Sub ArrangeMinimizedWindows()
  'VBA129
  Application.ArrangeMinimizedWindows
End Sub

See also

Application Object (Page 1909)
VBA Reference (Page 1745)

BackwardOneLevel Method

Description

Moves the selected objects one level backward within their current layer.

Syntax

Expression.BackwardOneLevel()
Expression
Necessary. An expression or element which returns an object of the "Selection" type.

Parameters
--

Example:
In the following example two objects are inserted in the active picture. The object inserted last is then moved backward one level:

Sub MoveObjectOneLevelBackward()
'VBA173
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
 .Top = 40
 .Left = 40
 .Selected = False
End With
With objRectangle
 .Top = 40
 .Left = 40
 .Width = 100
 .Height = 50
 .BackColor = RGB(255, 0, 255)
 .Selected = True
End With
MsgBox "Objects created and selected!"
ActiveDocument.Selection.BackwardOneLevel
End Sub

See also
SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)
BringToFront Method

Description
Brings the selected objects right to the front within their current layer.

Note
If the "BringToFront" method is used, the sequence of HMI objects can change in the HMIObjec.ts listing.

Syntax

\[ Expression.BringToFront() \]

Expression
Necessary. An expression or element which returns an object of the "Selection" type.

Parameters
--
Example:

In the following example two objects are inserted in the active picture. The object inserted last is then brought to the front:

Sub MoveObjectToFront()
'VBA198
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 40
  .Left = 40
  .Width = 100
  .Height = 50
  .BackColor = RGB(255, 0, 255)
  .Selected = False
End With
MsgBox "The objects circle and rectangle are created" & vbCrLf & "Only the circle is selected!"
ActiveDocument.Selection.BringToFront
MsgBox "The selection is moved to the front."
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

CascadeWindows Method

Description

Arranges all open pictures in the Graphics Designer in a cascade (i.e. overlapping).

syntax

Expression.Method(Parameter)

Expression

Necessary. An expression or element which returns an object of the "Application" type.
Parameters

--

Example:
In the following example all open pictures in the Graphics Designer are arranged in a cascade. For this example to work, you must have opened a number of pictures in the Graphics Designer:

```vba
Sub CascadeWindows()
'VBA130
Application.CascadeWindows
End Sub
```

See also

- VBA Reference (Page 1745)
- Application Object (Page 1909)

CenterHorizontally Method

Description
Using this method, the objects selected in the specified picture are centered horizontally.

Syntax

```
Expression.CenterHorizontally()
```

Expression
Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example two objects are inserted at different positions in the current picture and then centered horizontally:

Sub CenterSelectedObjectsHorizontally()
'VBA131
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
 .Top = 40
 .Left = 40
 .Selected = True
End With
With objRectangle
 .Top = 80
 .Left = 80
 .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.CenterHorizontally
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

CenterVertically Method

Description

Using this method, the objects selected in the specified picture are centered vertically.

Syntax

Expression.CenterVertically()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example two objects are inserted at different positions in the current picture and then centered vertically:

```vba
Sub CenterSelectedObjectsVertically()
'VBA132
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.CenterVertically
End Sub
```

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

CheckSyntax Method

Description

Checks whether the syntax of the specified C script is correct.
Use the CheckSyntax method in conjunction with the Compiled Property.

Syntax

```
Expression.CheckSyntax(CheckOK, Error)
```

Expression

Necessary. An expression or element which returns an object of the "DynamicDialog" type.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckOK (Boolean)</td>
<td>TRUE if the syntax of the specified C script is correct.</td>
</tr>
<tr>
<td>Error (String)</td>
<td>The message text that is output if the C script is incorrect.</td>
</tr>
</tbody>
</table>

Example:

```
--
```

See also

- DynamicDialog Object (Page 1955)
- VBA Reference (Page 1745)

Close Method

Description

Closes the specified picture and removes it from the document listing.

Note

Changes that have not been saved will be lost.

Syntax 1

```
Expression.Close(FileName)
```

Expression

Necessary. An expression or element which returns an object of the "Documents" type.

Syntax 2

```
Expression.Close()
```

Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName (String)</td>
<td>The name of the PDL file to be closed.</td>
</tr>
</tbody>
</table>
Example:

In the following example the picture "Test.PDL" will For this example to work, you must have opened the picture "Test.PDL":

Sub CloseDocumentUsingTheFileName()
  'VBA134
  Dim strFile As String
  strFile = Application.ApplicationDataPath & "test.pdl"
  Application.Documents.Close (strFile)
End Sub

In the following example the active picture in the Graphics Designer will be closed:
Sub CloseDocumentUsingActiveDocument()
  'VBA135
  ActiveDocument.Close
End Sub

See also

- Document Object (Page 1949)
- ActiveDocument Property (Page 2107)
- Documents Object (Listing) (Page 1952)
- VBA Reference (Page 1745)

CloseAll Method

Description
Closes all the pictures opened in the Graphics Designer and removes them from the documents listing.

Note
Changes that have not been saved will be lost.

Syntax

Expression.CloseAll()

Expression
Necessary. An expression or element which returns an object of the "Documents" type.

Parameters

--
Example:

In the following example all open pictures in the Graphics Designer are closed:

```vba
Sub CloseAllDocuments()
'VBA136
Application/Documents.CloseAll
End Sub
```

See also

- Documents Object (Listing) (Page 1952)
- VBA Reference (Page 1745)

ConvertToScript Method

Description

Converts the specified Dynamic dialog into a C script.
On conversion the associated DynamicDialog object is deleted.

Note

You cannot undo the conversion.

Syntax

```
Expression.ConvertToScript()
```

Expression

Necessary. An expression or element which returns an object of the "DynamicDialog" type.

Parameters

--
Example:

In the following example a circle will be inserted into the active picture and its radius will be dynamically configured using the Dynamic dialog. The Dynamic dialog will then be converted into a C script.

Sub ConvertDynamicDialogToScript()
    'VBA137
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    'Create dynamic
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
    "'NewDynamic1'")
    'configure dynamic. "ResultType" defines the valuerange-type:
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.Add 50, 40
        .AnalogResultInfos.Add 100, 80
        .AnalogResultInfos.ElseCase = 100
    End With
    MsgBox "The dynamic-dialog will be changed into a C-script."
    .ConvertToScript
End Sub

See also

DynamicDialog Object (Page 1955)
VBA Reference (Page 1745)

ConvertToVBScript method

Description

Converts the specified Dynamic dialog into a VB script.

On conversion, the associated DynamicDialog object is deleted.

Note

You cannot undo the conversion.

Syntax

Expression.ConvertToVBScript()
Expression

Required. An expression which returns an object of the "DynamicDialog" type.

Parameters

--

Example

In the following example, a circle is inserted into the active picture and its radius is dynamically configured using the Dynamic dialog. The Dynamic dialog is then converted into a VB script.

Sub ConvertDynamicDialogToVBScript()
'VBA908
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMIObject
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMIObject")
'Create dynamic
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
'configure dynamic. "ResultType" defines the valuerange-type:
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.Add 50, 40
  .AnalogResultInfos.Add 100, 80
  .AnalogResultInfos.ElseCase = 100
MsgBox "The dynamic-dialog will be changed into a VB-script."
  .ConvertToVBScript
End With
End Sub

See also

DynamicDialog Object (Page 1955)

ConvertWM method

Description

Is used internally for PowerCC.
CopySelection Method

Description
Using this method, the objects selected in the picture are copied to the clipboard.

syntax

Expression.CopySelection()

Expression
Necessary. An expression or element which returns an object of the "Document" or "Selection" type.

Parameters

Example:

In the following example two of the objects inserted in the active picture are selected. The selection is copied and pasted to a new picture:

Sub CopySelectionToNewDocument()
    'VBA138
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim iNewDoc As Integer
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    'Instead of "ActiveDocument.CopySelection" you can also write:
    '"ActiveDocument.Selection.CopySelection".
    ActiveDocument.CopySelection
    Application.Documents.Add hmiOpenDocumentTypeVisible
    iNewDoc = Application.Documents.Count
    Application.Documents(iNewDoc).PasteClipboard
End Sub
CopyToClipboard Method

Description
Copies an object from a folder in the Components Library to the clipboard.

Syntax

.Expression.CopyToClipboard()

Expression
Necessary. An expression or element which returns a FolderItem object of the "Item" type.

Parameters
--
Example:

In the following example the object "PC" from the "Global Library" will be copied into the folder "My Folder3" in the "Project Library":

```vba
Sub CopyObjectFromGlobalLibraryToProjectLibrary()
  'VBA139
  Dim objGlobalLib As HMISymbolLibrary
  Dim objProjectLib As HMISymbolLibrary
  Dim objFolderItem As HMIFolderItem

  Set objGlobalLib = Application.SymbolLibraries(1)
  Set objProjectLib = Application.SymbolLibraries(2)
  objProjectLib.FolderItems.AddFolder ("My Folder3")
  'copy object from "Global Library" to clipboard
  With objGlobalLib.
    .FolderItems(2).Folder.Item(2).Folder.Item(1).CopyToClipboard
  End With
  'paste object from clipboard into "Project Library"
  Set objFolderItem = objProjectLib.FindByDisplayName("My Folder3")
  objFolderItem.Folder.AddFromClipBoard ("Copy of PC/PLC")
End Sub
```

See also

SymbolLibrary Object (Page 2074)
FolderItem Object (Page 1972)
VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)

CreateCustomizedObject Method

Description

Creates a customized object from the objects selected in the specified picture. You then have to configure the customized object in the "Configuration Dialog".

For further information on this topic please refer to "Customized Objects" in this documentation and "Customized Object" in the WinCC documentation.

Syntax

```
Expression.CreateCustomizedObject()
```
**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

**Parameters**

--

**Example:**

In the following example two objects are inserted at different positions in the current picture and a customized object is then created:

```vba
Sub CreateCustomizedObject()
    'VBA140
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim objCustObject As HMIInternalizedObject
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects selected!"
    Set objCustObject = ActiveDocument.Selection.CreateCustomizedObject
    objCustObject.ObjectName = "myCustomizedObject"
End Sub
```

**See also**

SelectedObjects object (Listing) (Page 2060)

CustomizedObject Object (Page 1938)

VBA Reference (Page 1745)

Customized Objects (Page 1682)
CreateDynamicDialog method

Description
Dynamizing properties of pictures and objects depending on specific value ranges or variable statuses.

Syntax

Expression.CreateDynamicDialog([Code as String],iResultType as Long)

Expression
Required. An expression or element which returns an object of the "Property" type.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code (String)</td>
<td>Defines the function or tag that is used for dynamic purposes. Also specify the tag name in single quotation marks: &quot;Tag name&quot;</td>
</tr>
</tbody>
</table>
| iResultType (Long)   | Defines the type of value range:  
  - hmiResultTypeDirect = 0  
  - hmiResultTypeAnalog = 1  
  - hmiResultTypeBool = 2  
  - hmiResultTypeBit = 3 |

Example

In the following example the radius of a circle is given dynamics with the dynamic dialog. A tag name and a "ResultType" are assigned to the dynamic dialog.

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA820
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  'Create Object
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle","HMICircle")
  'Create dynamic (Tag "myTest" must be exist")
  Set objDynDialog = objCircle.Radius.CreateDynamicDialog("'myTest'"#,1)
End Sub

See also

FaceplateProperty object (Page 1971)
CreateDynamic Method

Description

Makes the specified property dynamic.

Syntax

Expression.CreateDynamic(DynamicType, [SourceCode])

Expression

Necessary. An expression or element which returns an object of the "Property" type.

Parameters

You only need use the "SourceCode" parameter if you want to make the specified property dynamic with the aid of the Dynamic dialog.

In all other types of dynamics you can omit the parameter.

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
</table>
| DynamicType (HMIDynamicCreationType) | Defines the type of dynamics:  
  - hmiDynamicCreationTypeVariableDirect: Dynamics with a tag  
  - hmiDynamicCreationTypeVariableIndirect: Dynamics with a tag In this type of dynamics you specify only the name of the tag whose value will be used for dynamic purposes.  
  - hmiDynamicCreationTypeScript: Dynamics with a script (C, VB).  
  - hmiDynamicCreationTypeDynamicDialog: Dynamizing with the dynamic dialog box: |
| SourceCode (String)         | Defines the function or tag that will be used for dynamic purposes.  
  Also specify the tag name in single quote marks: "Tag name" |
Example:

In this example a circle property "Top" will be made dynamic with the aid of the tag "NewDynamic":

Sub AddDynamicAsVariableDirectToProperty()
'VBA141
Dim objVariableTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("MyCircle", "HMICircle")
'Make property "Top" dynamic:
Set objVariableTrigger = objCircle.Top.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "NewDynamic")
' Define cycle-time
With objVariableTrigger
  .CycleType = hmiCycleType_2s
End With
End Sub

See also

Property Object (Page 2042)
DeleteDynamic Method (Page 1833)
VBA Reference (Page 1745)

CreateGroup Method

Description

Creates a group object from the objects selected in the specified picture.

For further information on this topic please refer to "Group Objects" in this documentation and "Group Object" in the WinCC documentation.

Syntax

Expression.CreateGroup()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example two objects are inserted at different positions in the current picture and a group object is then created:

Sub CreateGroup()
'VBA142
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
MsgBox "Objects selected!"
Set objGroup = ActiveDocument.Selection.CreateGroup
objGroup.ObjectName = "myGroup"
End Sub

See also

- SelectedObjects object (Listing) (Page 2060)
- Group Object (Page 1978)
- VBA Reference (Page 1745)
- Group Objects (Page 1674)

D-M

GetDeclutterObjectSize method

Description

Reads the limits displaying and hiding objects (decluttering) in the specified picture.

Syntax

\[ \text{Expression}.\text{GetDeclutterObjectSize}(\text{Min}, \text{Max}) \]
Expression

Required. An expression or element which returns an object of the "Document" type.

Parameter

<table>
<thead>
<tr>
<th>Parameter (data type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min (Long)</td>
<td>Lower size range in pixels.</td>
</tr>
<tr>
<td>Max (Long)</td>
<td>Upper size range in pixels.</td>
</tr>
</tbody>
</table>

Example

In the following example, the decluttering limits of the active picture are read and output:

```vba
Sub ReadSettingsOfPicture()
    Dim objectsize_min As Long, objectsize_max As Long
    ActiveDocument.GetDeclutterObjectSize objectsize_min, objectsize_max
    MsgBox objectsize_min & " " & objectsize_max
End Sub
```

Delete Method

Description

Deletes the specified object and removes it from the listing.

Syntax

```
Expression.Delete()
```

Expression

Necessary. An expression or element which returns objects of the following types.

- Assignment
- FolderItem
- LanguageText
- Menu
- MenuItem
- Object
- Toolbar
Parameters

Example:

In the following example the first object in the active picture will be deleted. For this example to work, you must have created at least one object in the active picture:

Sub ObjectDelete()
    'VBA143
    ActiveDocument.HMIObjects(1).Delete
End Sub

See also

LanguageText Object (Page 1999)
View Object (Page 2102)
VariableTrigger Object (Page 2099)
ToolBarItem Object (Page 2083)
FolderItem Object (Page 1972)
HMIObject Object (Page 1988)
MenuItem Object (Page 2013)
Menu Object (Page 2009)
VBA Reference (Page 1745)

DeleteAll Method

Description

Deletes all selected objects in the specified picture and removes them from the "Selection" and "HMIObjects" listings.

syntax

Expression.DeleteAll()
**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

**Parameters**

```
--
```

**Example:**

In the following example two objects are inserted at different positions in the current picture and then selected and deleted:

```vba
Sub DeleteAllSelectedObjects()
  'VBA145
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
  With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
  End With
  With objRectangle
    .Top = 80
    .Left = 80
    .Selected = True
  End With
  MsgBox "Objects selected!"
  ActiveDocument.Selection.DeleteAll
End Sub
```

**See also**

- SelectedObjects object (Listing) (Page 2060)
- VBA Reference (Page 1745)

**DeleteDynamic Method**

**Description**

Removes the dynamic characteristic from the specified property.

**syntax**

```
Expression.DeleteDynamic
```

---

*WinCC: Scripting (VBS, ANSI-C, VBA)*

*System Manual, 02/2016, A5E37536061-AA*
Expression
Necessary. An expression or element which returns an object of the "Property" type.

Parameters
--

Example:
In the following example the dynamic characteristic created with the aid of the CreateDynamic Method will be

Sub DeleteDynamicFromObjectMeinKreis()
    'VBA146
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects("MyCircle")
    objCircle.Top.DeleteDynamic
End Sub

See also
Property Object (Page 2042)
CreateDynamic Method (Page 1828)
VBA Reference (Page 1745)

DeselectAll Method

Description
Deselects all selected objects in the specified picture and removes them from the Selection listing.

Syntax
Expression.DeselectAll()

Expression
Necessary. An expression or element which returns an object of the "Selection" type.

Parameters
--
Example:

In the following example two objects are inserted at different positions in the current picture and selected. All selected objects are then deselected:

```vba
Sub SelectObjectsAndDeselectThemAgain()
    'VBA147
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 80
        .Selected = True
    End With
    MsgBox "Objects created and selected!"
    ActiveDocument.Selection.DeselectAll
    MsgBox "Objects deselected!"
End Sub
```

See also

SelectedObjects object (Listing) (Page 2060)

VBA Reference (Page 1745)

Destroy Method

Description

Ungroups the specified customized object. The objects remain intact.

Syntax

```vba
Expression_Destroy()
```

Expression

An expression or element which returns objects of the "CustomizedObject" types.

Parameters

--
Example:

An example showing how to use the Destroy Method can be found in this documentation under the heading "Editing a Customized Object with VBA".

See also

CustomizedObject Object (Page 1938)
Destroy Method (Page 1835)
Delete Method (Page 1831)
CreateCustomizedObject Method (Page 1825)
How to Edit a Customized Object with VBA (Page 1683)

DuplicateSelection Method

Description

Duplicates the objects selected in the specified picture. The objects created in this way are added to the HMIOObjects listing. The names of new objects are numbered consecutively with each duplication.

For instance if you duplicate an object called "Circle", the duplicate object is called "Circle1". If you duplicate the object called "Circle" once more, the resulting object is called "Circle2" and so on.

syntax

Expression.DivDuplicateSelection()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example two objects are inserted at different positions in the current picture and selected. They are then duplicated:

Sub DuplicateSelectedObjects()
'VBA149
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 80
  .Selected = True
End With
MsgBox "Objects created and selected!"
ActiveDocument.Selection.DuplicateSelection
End Sub

See also

- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- VBA Reference (Page 1745)

EvenlySpaceHorizontally Method

Description

Using this method, the objects selected in the specified picture are spaced horizontally at an even distance from one another.

syntax

Expression.EvenlySpaceHorizontally()
Parameters

Example:

In the following example three objects are inserted at different positions in the current picture and selected. They are then positioned horizontally at an even distance from one another:

Sub EvenlySpaceObjectsHorizontally()
'VBA150
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
    .Top = 30
    .Left = 0
    .Selected = True
End With
With objRectangle
    .Top = 80
    .Left = 42
    .Selected = True
End With
With objEllipse
    .Top = 48
    .Left = 162
    .BackColor = RGB(255, 0, 0)
    .Selected = True
End With
MsgBox "Objects created and selected!"
ActiveDocument.Selection.EvenlySpaceHorizontally
End Sub

See also

VBA Reference (Page 1745)
SelectedObjects object (Listing) (Page 2060)

EvenlySpaceVertically Method

Description

Using this method, the objects selected in the specified picture are spaced vertically at an even distance from one another.
syntax

Expression.EvenlySpaceVertically()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example three objects are inserted at different positions in the current picture and selected. They are then positioned vertically at an even distance from one another:

Sub EvenlySpaceObjectsVertically()
    'VBA151
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim objEllipse As HMIEllipse
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
    With objCircle
        .Top = 30
        .Left = 0
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 42
        .Selected = True
    End With
    With objEllipse
        .Top = 48
        .Left = 162
        .BackColor = RGB(255, 0, 0)
        .Selected = True
    End With
    MsgBox "Objects created and selected"
    ActiveDocument.Selection.EvenlySpaceVertically
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)
Export Method

Description

Saves the specified picture as an EMF file.

Syntax

\[ \text{Expression}.\text{Export}(\text{Type}, \text{Path}) \]

Expression

Required. An expression or element which returns an object of the "Document" type.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type (HMIImportExportType)</td>
<td>Defines the format in which the exported picture will be saved.</td>
</tr>
<tr>
<td>Path (String)</td>
<td>The path in which the picture is going to be exported. The path must exist.</td>
</tr>
</tbody>
</table>

Example

```
Sub ExportAllPicturesAsPDL()
'VBA152
Dim iPictureCounter As Integer
Dim strPath As String

strPath = "C:\WinCC_PDL_Export\"

'Count Pictures in Graphics Designer...
For iPictureCounter = 1 To grafexe/Documents.Count
    '...and export each picture as PDL-file to specified path:
    grafexe/Documents(iPictureCounter).Export hmiImportExportTypePDL, strPath
Next iPictureCounter
End Sub
```

See also

View Object (Page 2102)
Document Object (Page 1949)
Find Method

Description

Searches for objects in the specified picture and returns the search result as a collection object. You can search for the following object properties:

- Type
- Name
- Property

Syntax

Expression.Find([ObjectType], [ObjectName], [PropertyName])

Expression

Necessary. An expression or element which returns an object of the "HMIOBJECTS" type.

Parameters

You must specify at least one of the three parameters.

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectType (String)</td>
<td>The object type that is to be searched for. Specify the &quot;ProgID&quot; of the object concerned. &quot;Obtain the &quot;ProgID&quot; by prefixing the VBA object name with &quot;HMI&quot; &quot;(e.g. HMICircle or HMIRectangle)</td>
</tr>
<tr>
<td>ObjectName (String)</td>
<td>The name of the object that is to be searched for. You can use placeholders (?,*) in the object name in order to find objects with similar names.</td>
</tr>
<tr>
<td>PropertyName (String)</td>
<td>The name of the object property that is to be searched for. Specify the VBA property name concerned (e.g. &quot;BackColor&quot; in place of &quot;BackgroundColor&quot;).</td>
</tr>
</tbody>
</table>
Example:

In the following example, objects of the "HMICircle" type will be searched for in the active picture and the search result will be output:

```vba
Sub FindObjectsByType()
'VBA153
Dim colSearchResults As HMICollection
Dim objMember As HMIObject
Dim iResult As Integer
Dim strName As String
Set colSearchResults = ActiveDocument.HMIObjects.Find(ObjectType:="HMICircle")
For Each objMember In colSearchResults
iResult = colSearchResults.Count
strName = objMember.ObjectName
MsgBox "Found: " & CStr(iResult) & vbCrLf & "objectname: " & strName)
Next objMember
End Sub
```

Note

Further information on using the Find Method can be found in this documentation under the heading "Editing Standard Objects, Smart Objects and Windows Objects".

See also

- Type Property (Page 2450)
- Name Property (Page 2343)
- Property Object (Page 2042)
- HMIObjects Object (Listing) (Page 1990)
- How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1664)
- VBA Reference (Page 1745)

FindByDisplayName Method

Description

Searches the entire Components Library for the specified object. A FolderItem object is returned as the search result.

Note

The display name of the object is language-dependent. Only the language currently set will be taken into account when searching. The search ends with the first object found.
syntax

Expression.FindByDisplayName(DisplayName)

Expression

Necessary. An expression or element which returns an object of the "SymbolLibrary" type or the "FolderItems" listing.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayName (String)</td>
<td>The display name of the object that is to be searched for in the Components Library.</td>
</tr>
</tbody>
</table>

Example:

In the following example the entire library will be searched for the object "PC" and its display name will be output:

Sub FindObjectInSymbolLibrary()
"VBA154
Dim objGlobalLib As HMISymbolLibrary
Dim objFItem As HMIFolderItem
Set objGlobalLib = Application.SymbolLibraries(1)
Set objFItem = objGlobalLib.FindByDisplayName("PC")
MsgBox objFItem.DisplayName
End Sub

See also

FolderItem Object (Page 1972)
Accessing the component library with VBA (Page 1646)

FireConnectionEvents method

Description

Is used internally by the Graphics Designer.

FlipHorizontally Method

Description

Mirrors the selected objects in the specified picture along the horizontal midline.
The object type determines whether it is allowed to be mirrored (for instance an OLE Element cannot be mirrored). The properties are appropriately modified when mirroring is performed. For example, if you mirror an object of the "StaticText" type along the horizontal midline, the value of the "AlignmentTop" property changes from "0" to "2".

**Syntax**

```
Expression.FlipHorizontally()
```

**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

**Parameters**

--

**Example:**

In the following example a StaticText object will be inserted into the active picture and mirrored along the horizontal midline:

```vba
Sub FlipObjectHorizontally()
'VBA155
Dim objStaticText As HMIStaticText
Dim strPropertyName As String
Dim iPropertyValue As Integer
Set objStaticText = ActiveDocument.HMIObjects.AddHMIObject("Textfield", "HMIStaticText")
strPropertyName = objStaticText.Properties("Text").Name
With objStaticText
    .Width = 120
    .Text = "Sample Text"
    .Selected = True
    iPropertyValue = .AlignmentTop
    MsgBox "Value of '" & strPropertyName & '" before flip: " & iPropertyValue
    ActiveDocument.Selection.FlipHorizontally
    iPropertyValue = objStaticText.AlignmentTop
    MsgBox "Value of '" & strPropertyName & '" after flip: " & iPropertyValue
End With
End Sub
```

**See also**

SelectedObjects object (Listing) (Page 2060)

VBA Reference (Page 1745)
FlipVertically Method

Description

Mirrors the selected objects in the specified picture along the vertical midline.

The object type determines whether it is allowed to be mirrored (for instance an OLE Element cannot be mirrored). The properties are appropriately modified when mirroring is performed. For example if you mirror an object of the "StaticText" type along the vertical midline, the value of the "AlignmentLeft" property changes from "0" to "2".

Syntax

Expression.FlipVertically()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example a StaticText object will be inserted into the active picture and mirrored along the vertical midline:

```vba
Sub FlipObjectVertically()
    'VBA156
    Dim objStaticText As HMIStaticText
    Dim strPropertyName As String
    Dim iPropertyValue As Integer
    Set objStaticText = ActiveDocument.HMIObjects.AddHMIObject("Textfield", "HMIStaticText")
    strPropertyName = objStaticText.Properties("Text").Name
    With objStaticText
        .Width = 120
        .Text = "Sample Text"
        .Selected = True
        .AlignmentLeft = 0
        iPropertyValue = .AlignmentLeft
        MsgBox "Value of " & strPropertyName & " before flip: " & iPropertyValue
    End With
    ActiveDocument.Selection.FlipVertically
    iPropertyValue = objStaticText.AlignmentLeft
    MsgBox "Value of " & strPropertyName & " after flip: " & iPropertyValue
End Sub
```
**ForwardOneLevel Method**

**Description**

Moves the selected objects one level forward within their current layer.

**syntax**

```
Expression.ForwardOneLevel()
```

**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

**Parameters**

--

**Example:**

In the following example two objects are inserted in the active picture. The object inserted first is then moved forward one level:

```vba
Sub MoveObjectOneLevelForward()
    'VBA174
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIOBJECT("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIOBJECT("sRectangle", "HMIRectangle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
    End With
    With objRectangle
        .Top = 40
        .Left = 40
        .Width = 100
        .Height = 50
        .BackColor = RGB(255, 0, 255)
        .Selected = False
    End With
    MsgBox "Objects created and selected!"
    ActiveDocument.Selection.ForwardOneLevel
End Sub
```
See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

GetItemByPath Method

Description

Returns a FolderItem object (folder or object) located on the specified internal access path in the Components Library.

Note

To obtain the internal access path, select the "Copy Path" command from The internal access path to the folder or object will then be copied to the clipboard.

syntax

Expression.GetItemByPath(PathName)

Expression

Necessary. An expression or element which returns an object of the "SymbolLibrary" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathName (String)</td>
<td>The internal access path on which the object is located in the Components Library.</td>
</tr>
</tbody>
</table>

Example:

In this example one object from the entire library will be returned and its display name will be output:

```vba
Sub ShowDisplayName()
    'VBA157
    Dim objGlobalLib As HMISymbolLibrary
    Dim objFItem As HMIFolderItem
    Set objGlobalLib = Application.SymbolLibraries(1)
    Set objFItem = objGlobalLib.GetItemByPath("\Folder1\Folder2\Object1")
    MsgBox objFItem.DisplayName
End Sub
```
See also
SymbolLibrary Object (Page 2074)
FolderItem Object (Page 1972)
Accessing the component library with VBA (Page 1646)

InsertFromMenuItem Method

Description
Inserts into an existing, user-defined toolbar a new icon that references an existing menu entry in a user-defined menu.
Use this method if you wish to set up a toolbar so that it contains the same commands as an existing user-defined menu.

Syntax
Expression.InsertFromMenuEntry(Position, Key, pMenuItem, DefaultToolTipText)

Expression
Required. An expression or element which returns an object of the "ToolbarItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the icon within the user-defined toolbar.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the symbol. Use unique names for &quot;Key&quot; <em>(e.g. ItItem1_1)</em>.</td>
</tr>
<tr>
<td>pMenuItem (HIMMenuItems)</td>
<td>The MenuItem object that is intended to be referenced.</td>
</tr>
<tr>
<td>DefaultToolTipText (String)</td>
<td>Defines for the icon concerned the tool tip text that will be displayed when you move the mouse over the icon.</td>
</tr>
</tbody>
</table>
Example:

In this example a user-defined menu and a user-defined toolbar will be inserted in the active picture. The icon calls up the menu entry "Hello World" from the user-defined menu:

```
Sub ToolBarItem_InsertFromMenuItem()
'VBA158
Dim objMenu As HMIMenu
Dim objToolbarItem As HMIToolbarItem
Dim objToolbar As HMIToolbar
Dim objMenuItem As HMIMenuItem
Set objMenu = Application.CustomMenus.InsertMenu(1, "Menu1", "TestMenu")
'*
'* Note:
'* The object-reference has to be unique.
'********************************************************************************
',
Set objMenuItem = Application.CustomMenus(1).MenuItems.InsertMenuItem(1, "MenuItem1", "Hello World")
Application.CustomMenus(1).MenuItems(1).Macro = "HelloWorld"
Set objToolbar = Application.CustomToolbars.Add("Toolbar1")
Set objToolBarItem = Application.CustomToolbars(1).ToolBarItems.InsertFromMenuItem(1, "ToolbarItem1", objMenuItem, "Call's Hello World of TestMenu")
End Sub

Sub HelloWorld()
MsgBox "Procedure 'HelloWorld()' is execute."
End Sub
```

See also

- ToolBarItem Object (Listing) (Page 2085)
- InsertSeparator Method (Page 1853)
- Add Method (CustomToolbars Listing) (Page 1788)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
**InsertMenu Method**

**Description**

Creates a new, user-defined menu. There is a difference between application-specific and picture-specific user-defined menus:

- Application-specific menu: This is linked to the Graphics Designer and is also only visible when all the pictures in the Graphics Designer are closed. "Place the VBA code in the document called "GlobalTemplateDocument" or "ProjectTemplateDocument" and use the Application property.

- Picture-specific menu: Is linked with a specific picture and remains visible as long as the picture is visible. Place the VBA code in the document called "ThisDocument" for the desired picture and use the ActiveDocument property.

**Syntax**

```
Expression.InsertMenu(Position, Key, DefaultLabel)
```

**Expression**

Necessary. An expression or element which returns an object of the "CustomMenus" type.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the user-defined menu within the menu bar. However, picture-specific menus are always positioned to the right of application-specific menus.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the user-defined menu. Use unique names for &quot;Key&quot; (e.g. &quot;DocMenu1&quot;)</td>
</tr>
<tr>
<td>DefaultLabel (String)</td>
<td>The name of the user-defined menu.</td>
</tr>
</tbody>
</table>
Example:

In the following example, a user-defined menu with two menus entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line:

Sub CreateDocumentMenus()
'VBA159
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Dim objSubMenu As HMIMenuItem

Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "First MenuItem")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "Second MenuItem")

'Insert a dividing rule into customized menu:
Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")

Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "First SubMenu")

Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "First item in sub-menu")
Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "Second item in sub-menu")

End Sub

The diagram shows the generated menu structure.

See also

Menus Object (Listing) (Page 2011)
InsertSubmenu Method (Page 1854)
InsertSeparator Method (Page 1853)
InsertMenuItem Method (Page 1852)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)
InsertMenuItem Method

Description

Inserts a new entry in a user-defined menu.

Syntax

Expression.InsertMenuItem(Position, Key, DefaultLabel)

Expression

Necessary. An expression or element which returns an object of the "MenuItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the submenu within the user-defined menu.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the submenu. Use unique names for &quot;Key&quot; (e.g. dSubMenu1_4).</td>
</tr>
<tr>
<td>DefaultLabel (String)</td>
<td>Defines the name of the submenu.</td>
</tr>
</tbody>
</table>

Example:

In the following example, a user-defined menu with two menus entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line:

Sub CreateDocumentMenus()
  'VBA160
  Dim objDocMenu As HMIMenu
  Dim objMenuItem As HMIMenuItem
  Dim objSubMenu As HMIMenuItem
  Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "First MenuItem")
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "Second MenuItem")
  'Insert a dividing rule into customized menu:
  Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")
  'Insert a submenu:
  Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "First SubMenu")
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "First item in submenu")
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "Second item in submenu")
End Sub
The diagram shows the menu structure.

<table>
<thead>
<tr>
<th>Doc_Menu_1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>First MenuItem</td>
<td></td>
</tr>
<tr>
<td>Second MenuItem</td>
<td></td>
</tr>
<tr>
<td>First SubMenu</td>
<td></td>
</tr>
<tr>
<td>First item in sub-menu</td>
<td></td>
</tr>
<tr>
<td>Second item in sub-menu</td>
<td></td>
</tr>
</tbody>
</table>

See also

- MenuItems Object (Listing) (Page 2015)
- MenuItem Object (Page 2013)
- InsertSubmenu Method (Page 1854)
- InsertSeparator Method (Page 1853)
- InsertMenu Method (Page 1850)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)

InsertSeparator Method

Description

Inserts a dividing line in a user-defined menu or user-defined toolbar.

Syntax

```
Expression.InsertSeparator(Position, Key)
```

Expression

Necessary. An expression or element which returns an object of the "MenuItems" or "ToolbarItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the dividing line within the user-defined menu or user-defined toolbar.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the dividing line. Use unique names for &quot;Key&quot; (e.g. &quot;tSeparator1_2&quot;).</td>
</tr>
</tbody>
</table>
Example:

In the following example a user-defined toolbar with two icons is created in the active picture. These icons are separated by a dividing line:

Sub AddDocumentSpecificCustomToolbar()
    'VBA161
    Dim objToolbar As HMIToolbar
    Dim objToolbarItem As HMIToolbarItem
    Set objToolbar = ActiveDocument.CustomToolbars.Add("DocToolbar")
        'Add toolbar-item to userdefined toolbar
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "tItem1_1", "First symbol-icon")
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(3, "tItem1_3", "Second symbol-icon")
        'Insert dividing rule between first and second symbol-icon
    Set objToolbarItem = objToolbar.ToolbarItems.InsertSeparator(2, "tSeparator1_2")
End Sub

See also

ToolbarItems Object (Listing) (Page 2085)
MenuItems Object (Listing) (Page 2015)
InsertToolbarItem Method (Page 1856)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)

InsertSubmenu Method

Description

Inserts a submenu into an existing user-defined menu.

Syntax

Expression.InsertSubmenu(Position, Key, DefaultLabel)

Expression

Necessary. An expression or element which returns an object of the "MenuItem" type
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the submenu within the user-defined menu.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the submenu. Use unique names for &quot;Key&quot; (e.g. dSubMenu1_4).</td>
</tr>
<tr>
<td>DefaultLabel (String)</td>
<td>Defines the name of the submenu.</td>
</tr>
</tbody>
</table>

Example:

In the following example, a user-defined menu with two menu entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line:

```
Sub CreateDocumentMenus()
    'VBA162
    Dim objDocMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem
    Dim objSubMenu As HMIMenuItem

    Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
    Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "First MenuItem")
    Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "Second MenuItem")
    Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")
    Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "First SubMenu")
    Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "First item in submenu")
    Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "Second item in submenu")
End Sub
```

The diagram shows the menu structure:
InsertToolbarItem Method

Description

Inserts a new icon in an existing user-defined toolbar.

Syntax

Syntax:  
Expression.InsertToolbarItem(Position, Key, DefaultToolTipText)

Expression
Necessary. An expression or element which returns an object of the "ToolbarItems" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position (Long)</td>
<td>Defines the position of the icon within the user-defined toolbar.</td>
</tr>
<tr>
<td>Key (Variant)</td>
<td>Identifies the symbol. Use unique names for &quot;Key&quot; &quot;(e.g. tItem1_1).</td>
</tr>
<tr>
<td>DefaultToolTipText (String)</td>
<td>Defines for the icon concerned the tool tip text that will be displayed when you move the mouse over the icon.</td>
</tr>
</tbody>
</table>
Example:

In the following example a user-defined toolbar with two icons is created in the active picture. These icons are separated by a dividing line:

Sub AddDocumentSpecificCustomToolbar()
'VBA163
Dim objToolbar As HMIToolbar
Dim objToolbarItem As HMIToolbarItem
Set objToolbar = ActiveDocument.CustomToolbars.Add("DocToolbar")
'Add toolbar-item to userdefined toolbar
Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "tItem1_1", "First symbol-icon")
Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(3, "tItem1_3", "Second symbol-icon")
'Insert dividing rule between first and second symbol-icon
Set objToolbarItem = objToolbar.ToolbarItems.InsertSeparator(2, "tSeparator1_2")
End Sub

See also

ToolbarItems Object (Listing) (Page 2085)
InsertSeparator Method (Page 1853)
Add Method (CustomToolbars Listing) (Page 1788)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)

IsCSLayerVisible Method

Description

Returns TRUE if the specified CS layer is visible.

syntax

Expression.IsCSLayerVisible(Index)
**Expression**

Necessary. An expression or element which returns an object of the "Document" type.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>Defines the CS layer. Value range from 1 to 32. Layer0 corresponds to the index value &quot;1&quot;.</td>
</tr>
</tbody>
</table>

**Example:**

The following example determines whether CS layer 1 in the copy of the active picture is visible and outputs the result:

```vba
Sub IsCSLayerVisible()
    'VBA164
    Dim objView As HMIView
    Dim strLayerName As String
    Dim iLayerIdx As Integer
    Set objView = ActiveDocument.Views(1)
    objView.Activate
    iLayerIdx = 2
    strLayerName = ActiveDocument.Layers(iLayerIdx).Name
    If objView.IsCSLayerVisible(iLayerIdx) = True Then
        MsgBox "CS " & strLayerName & " is visible"
    Else
        MsgBox "CS " & strLayerName & " is invisible"
    End If
End Sub
```

**See also**

- Document Object (Page 1949)
- VBA Reference (Page 1745)
- Editing Layers with VBA (Page 1657)

**IsRTLayerVisible Method**

**Description**

Returns TRUE if the specified RT layer is visible.

**syntax**

`Expression.IsRTLayerVisible(Index)`
Expression
Necessary. An expression or element which returns an object of the "Document" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>Defines the RT layer. Value range from 1 to 32.</td>
</tr>
<tr>
<td></td>
<td>Layer0 corresponds to the index value &quot;1&quot;.</td>
</tr>
</tbody>
</table>

Example:

The following example determines whether RT layer 1 is visible and outputs the result:

```vba
Sub RTLayerVisibility()
  'VBA165
  Dim strLayerName As String
  Dim iLayerIdx As Integer
  iLayerIdx = 2
  strLayerName = ActiveDocument.Layers(iLayerIdx).Name
  If ActiveDocument.IsRTLayerVisible(iLayerIdx) = True Then
    MsgBox "RT " & strLayerName & " is visible"
  Else
    MsgBox "RT " & strLayerName & " is invisible"
  End If
End Sub
```

See also

Document Object (Page 1949)
VBA Reference (Page 1745)
Editing Layers with VBA (Page 1657)

Item Method

Description

Returns an element from a listing.

Syntax

```vba
Expression.Item(Index)
```

Expression

Necessary. An expression or element which returns an object.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>The name or index number of an element from the listing. You can use the Object Name as the name. As the index number you can use a numerical expression (from 1 up to the value of the Count property of the listing). If the entered value fails to match any element in the listing, this counts as an error.</td>
</tr>
</tbody>
</table>

Example:

Note
The Item Method is the default method for listings. Both the following examples give the same result.

In the following example the name of the first picture in the Graphics Designer is output:

```
Sub ShowDocumentNameLongVersion()
  'VBA166
  Dim strDocName As String
  strDocName = Application.Documents.Item(3).Name
  MsgBox strDocName
  End Sub

Sub ShowDocumentNameShortVersion()
  'VBA167
  Dim strDocName As String
  strDocName = Application.Documents(3).Name
  MsgBox strDocName
  End Sub
```

See also

- VariableStateValues Object (Listing) (Page 2097)
- Count Property (Page 2201)
- Views Object (Listing) (Page 2103)
- VariableTriggers Object (Listing) (Page 2101)
- ToolbarItems Object (Listing) (Page 2085)
- Toolbars Object (Listing) (Page 2081)
- SymbolLibraries Object (Listing) (Page 2073)
- SelectedObjects object (Listing) (Page 2060)
ItemByLcid Method

Description

Selects the language for which you wish to enter the font settings. Read only access.

Note

You can only select languages in which you have already configured.

Syntax

Expression.ItemByLcid(LangID)

Expression

Required. An expression or element which returns an object of the "LanguageFonts" type.
Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LangID (Long)</td>
<td>This is the language identifier. The list of language identifiers is contained, for example, in the &quot;Languages.csv&quot; file that is found in the index of the WinCC documentation.</td>
</tr>
</tbody>
</table>

Example

The following example sets the font attributes of a button for French and English. In contrast to English, French is displayed on the button in a smaller font with a constant tracking (Courier New, 12pt):

```vba
Sub ExampleForLanguageFonts()
    'VBA168
    Dim objLangFonts As HMI LanguageFonts
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    objButton.Text = "Hello"
    Set objLangFonts = objButton.LDFonts
    'To make font settings for English:
    With objLangFonts.ItemByLCID(1033)
        .Family = "Times New Roman"
        .Bold = False
        .Italic = True
        .Underlined = False
        .Size = 14
    End With
    'To make font settings for French:
    With objLangFonts.ItemByLCID(1036)
        .Family = "Courier New"
        .Bold = True
        .Italic = False
        .Underlined = True
        .Size = 12
    End With
End Sub
```

See also

LanguageFonts Object (Listing) (Page 1997)
LoadDefaultConfig Method

Description
Loads the file in which the default settings for objects are saved. The PDD file is located in the "GraCS" folder of the current project.

Syntax
Expression.LoadDefaultConfig(FileName)

Expression
Necessary. An expression or element which returns an object of the "Application" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName (String)</td>
<td>The name of the PDD file which it is intended to load.</td>
</tr>
</tbody>
</table>

Example:
In the following example the file "Test.PDD" will be loaded. For this example to work, you must have previously saved the file. You can do this with the aid of the SaveDefaultConfig Method:

```vba
Sub LoadDefaultConfig()
'VBA169
Application.LoadDefaultConfig("Test.PDD")
End Sub
```

See also
Application Object (Page 1909)
SaveDefaultConfig Method (Page 1881)
VBA Reference (Page 1745)

MoveOneLayerDown Method

Description
Moves the selected object in the specified picture into the next lowest layer.
syntax

Expression.MoveOneLayerDown()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example a circle in the active picture is inserted in the third layer and then moved to the next lowest layer:

Sub MoveObjectOneLayerDown()
    'VBA170
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    With objCircle
        .Top = 40
        .Left = 40
        .Selected = True
        .Layer = 3
        MsgBox "Circle is inserted into layer" & Str(.Layer)
        ActiveDocument.Selection.MoveOneLayerDown
        MsgBox "Circle is moved into layer" & Str(.Layer)
    End With
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

MoveOneLayerUp Method

Description

Moves the selected object in the specified picture into the next highest layer.

syntax

Expression.MoveOneLayerUp()
Expression
Necessary. An expression or element which returns an object of the "Selection" type.

Parameters
--

Example:

In the following example a circle in the active picture is inserted in the third layer and then moved to the next highest layer:

```
Sub MoveObjectOneLayerUp()
  'VBA171
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
  With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
    .Layer = 3
    MsgBox "Circle is inserted into layer" & Str(.Layer)
    ActiveDocument.Selection.MoveOneLayerUp
    MsgBox "Circle is moved into layer" & Str(.Layer)
  End With
End Sub
```

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)

MoveSelection Method

Description
Moves one or more objects selected in the picture by the specified coordinates.

Note
When you want to reposition one or more selected objects, use the properties "Left" and "Top".

Syntax

```
Expression.MoveSelection(PosX, PosY)
```

Expression
Required. An expression or element which returns an object of the "Document" or "Selection" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PosX (Long)</td>
<td>The number of pixels by which the selection is to be moved horizontally.</td>
</tr>
<tr>
<td>PosY (Long)</td>
<td>The number of pixels by which the selection is to be moved vertically.</td>
</tr>
</tbody>
</table>

Example:
In the following example two objects are inserted at different positions in the current picture and selected. The selection is then moved 30 pixels to the right and 40 pixels down:

Sub MoveSelectionToNewPostion()
'VBA172
Dim nPosX As Long
Dim nPosY As Long
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
End With
With objRectangle
    .Top = 80
    .Left = 80
    .Selected = True
End With
MsgBox "Objects selected!"
nPosX = 30
nPosY = 40
ActiveDocument.MoveSelection nPosX, nPosY
End Sub

See also
Top Property (Page 2446)
Left Property (Page 2305)
Document Object (Page 1949)
VBA Reference (Page 1745)
Open Method

Description

Opens an existing picture in the Graphics Designer and adds it to the documents listing.

Syntax

Expression.Open(FileName, [HMIOpenDocumentType])

Expression

Necessary. An expression or element which returns an object of the "Documents" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName (String)</td>
<td>The name of the PDL file to be opened. Unless you saved the PDL file in the &quot;GraCS&quot; folder of the open project, you must also specify the path at the same time.</td>
</tr>
<tr>
<td>HMIOpenDocumentType (HMIDocumentType)</td>
<td>Defines how the picture will be opened:</td>
</tr>
<tr>
<td></td>
<td>- HMIDocumentTypeVisible: Opens the picture for direct processing. This is the default setting if you do not specify the parameter.</td>
</tr>
<tr>
<td></td>
<td>- HMIDocumentTypeInvisible: Opens the picture in invisible mode, i.e. it is not displayed in the Graphics Designer. You can only address the picture via the Documents listing, and make it visible again by means of the Hide property.</td>
</tr>
</tbody>
</table>

Example:

In the following example the picture "Test" will be opened. For this example to work, you must have previously saved a picture with the name "Test" in the "GraCS" folder of the open project.

Sub OpenDocument()
    'VBA175
    Application.Documents.Open "Test.PDL", hmiOpenDocumentTypeVisible
End Sub
PasteClipboard Method

Description

Pastes the contents of the clipboard into the specified picture.

Note

The clipboard must contain objects from the Graphics Designer. Other contents (such as ASCII text) will not be pasted.

Syntax

Expression.PasteClipboard()

Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

--

Example:

In the following example all the objects selected in the active picture are copied to the clipboard and then pasted into a new picture. For this example to work, you must have selected at least one object in the active picture:

```vba
Sub CopySelectionToNewDocument()
    'VBA176
    Dim iNewDoc As String
    Application.Documents.CopySelection
    Application.Documents.Add hmiOpenDocumentTypeVisible
    iNewDoc = Application.Documents.Count
    Application.Documents(iNewDoc).PasteClipboard
End Sub
```
PrintDocument Method

Description

Prints the specified copy of the picture using the current printer settings.

Syntax

Expression.PrintDocument()

Expression

Necessary. An expression or element which returns an object of the "View" type.

Parameters

--

Example:

In the following example a copy of the active picture is created and then activated and printed:

Sub CreateAndPrintView()
    'VBA177
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
    objView.PrintDocument
End Sub

See also

View Object (Page 2102)
VBA Reference (Page 1745)
PrintProjectDocumentation Method

Description

Prints out the project documentation for the current picture complete with all the objects it contains and their properties via the reporting system in WinCC (Report Designer).

You must first have set the print settings (such as page range) in the "Print Job Properties" dialog. To do this, go to the Graphics Designer and select the menu command "File" > "Project Documentation - Setup".

---

Note

The project documentation will be output on the printer that was set up in the Report Designer. You can design the print layout to suit your needs with the aid of the Report Designer.

---

Syntax

Expression.PrintProjectDocumentation()

Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

---

Example:

In the following example the project documentation for the active picture will be printed:

Sub ToPrintProjectDocumentation()
 'VBA178
 ActiveDocument.PrintProjectDocumentation
 End Sub

See also

Document Object (Page 1949)

VBA Reference (Page 1745)
Remove Method

Description

Removes an object from a selection of objects or from a group object.

Syntax

Expression.Remove(Index)

Expression

Necessary. An expression or element which returns an object of the "GroupedObjects" or "Selection" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>The name or index number of the object that is intended to be removed. You can use the Object Name as the name. As the index number you can use a numerical expression (from 1 up to the value of the Count property of the listing). If the entered value fails to match any element in the listing, this counts as an error.</td>
</tr>
</tbody>
</table>
Example:

In the following example three objects will first be inserted in the active picture and selected. Then one object will be removed from the selection and the remaining objects will be grouped. Then the first object will be removed from the group object:

Sub RemoveObjectFromGroup()
'VBA179
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
  .Top = 30
  .Left = 0
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 42
  .Selected = True
End With
With objEllipse
  .Top = 48
  .Left = 162
  .Width = 40
  .BackColor = RGB(255, 0, 0)
  .Selected = True
End With
MsgBox "Objects selected!"
Set objGroup = ActiveDocument.Selection.CreateGroup
MsgBox "Group-object is created."
objGroup.GroupedHMIObjects.Remove ("sEllipse")
MsgBox "The ellipse is removed from group-object."
End Sub

See also
- SelectedObjects object (Listing) (Page 2060)
- GroupedObjects Object (Listing) (Page 1983)
- VBA Reference (Page 1745)

Rotate Method

Description

Rotates the object selected in the specified picture by 90° clockwise.
syntax

Expression.Rotate()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects will be inserted in the active picture and then grouped. The group object will then be rotated once:

Sub RotateGroupObject()
    'VBA180
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim objGroup As HMIGroup
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    With objRectangle
        .Top = 30
        .Left = 30
        .Width = 80
        .Height = 40
        .Selected = True
    End With
    With objCircle
        .Top = 30
        .Left = 30
        .BackColor = RGB(255, 255, 255)
        .Selected = True
    End With
    MsgBox "Objects selected!"
    Set objGroup = ActiveDocument.Selection.CreateGroup
    MsgBox "Group-object created."
    objGroup.Selected = True
    ActiveDocument.Selection.Rotate
End Sub

See also

VBA Reference (Page 1745)
SelectedObjects object (Listing) (Page 2060)
SameHeight Method

Description

Sets the "Height" property for all selected objects in the specified picture to the smallest available value.

Syntax

Expression. SameHeight()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example three objects of different sizes will be inserted in the active picture. Then all objects will be selected and set to the same height:

```vba
Sub ApplySameHeightToSelectedObjects()
'VBA181
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
    .Top = 30
    .Left = 0
    .Height = 15
    .Selected = True
End With
With objRectangle
    .Top = 80
    .Left = 42
    .Height = 40
    .Selected = True
End With
With objEllipse
    .Top = 48
    .Left = 162
    .Width = 40
    .Height = 120
    .BackColor = RGB(255, 0, 0)
    .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.SameHeight
End Sub
```

See also

- Height Property (Page 2269)
- SelectedObjects object (Listing) (Page 2060)
- VBA Reference (Page 1745)
SameWidth Method

Description

Sets the "Width" property for all selected objects in the specified picture to the smallest available value.

Syntax

Expression.SameWidth()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

---
Example:

In the following example three objects of different sizes will be inserted in the active picture. Then all objects will be selected and set to the same width:

```vba
Sub ApplySameWidthToSelectedObjects()
'VBA182
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
  .Top = 30
  .Left = 0
  .Width = 15
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 42
  .Width = 40
  .Selected = True
End With
With objEllipse
  .Top = 48
  .Left = 162
  .Width = 120
  .BackColor = RGB(255, 0, 0)
  .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.SameWidth
End Sub
```

See also

- Width Property (Page 2553)
- SelectedObjects object (Listing) (Page 2060)
- VBA Reference (Page 1745)

SameWidthAndHeight Method

Description

Sets the "Height" and "Width" properties for all selected objects in the specified picture to the smallest available value.
syntax

\[ \text{Expression}.\text{SameWidthAndHeight}() \]

**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

**Parameters**

--

**Example:**

In the following example three objects of different sizes will be inserted in the active picture. Then all objects will be selected and set to the same height:

```vba
Sub ApplySameWidthAndHeightToSelectedObjects()
    'VBA183
    Dim objCircle As HMICircle
    Dim objRectangle As HMIRectangle
    Dim objEllipse As HMIEllipse
    Dim objGroup As HMIGroup
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
    Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
    With objCircle
        .Top = 30
        .Left = 0
        .Height = 15
        .Selected = True
    End With
    With objRectangle
        .Top = 80
        .Left = 42
        .Width = 25
        .Height = 40
        .Selected = True
    End With
    With objEllipse
        .Top = 48
        .Left = 162
        .Width = 40
        .Height = 120
        .BackColor = RGB(255, 0, 0)
        .Selected = True
    End With
    MsgBox "Objects selected!"
    ActiveDocument.Selection.SameWidthAndHeight
End Sub
```
Save Method

Description
Saves the specified picture under its current name.

Syntax

Expression.Save()

Expression
Necessary. An expression or element which returns an object of the "Document" type.

Parameters
--

Example:
In the following example the active picture in the Graphics Designer will be saved:

Sub SaveDocument()
    'VBA184
    ActiveDocument.Save
End Sub

See also
ActiveDocument Property (Page 2107)
Document Object (Page 1949)
VBA Reference (Page 1745)
SaveAll Method

Description

Saves all the open pictures in the Graphics Designer under their current names.

Syntax

```vba
Expression.SaveAll()
```

Expression

Necessary. An expression or element which returns an object of the "Documents" type.

Parameters

--

Example:

In the following example all open pictures in the Graphics Designer are saved:

```vba
Sub SaveAllDocuments()
  'VBA185
  Application/Documents.SaveAll
End Sub
```

See also

Documents Object (Listing) (Page 1952)
VBA Reference (Page 1745)

SaveAs Method

Description

Saves the specified picture under a new name.

If a previously existing picture is to be overwritten, it must be ascertained prior to the SaveAs method call that this picture is permitted to be overwritten. You must inquire the LockedByCreatorID property of the picture to be overwritten to do so. Otherwise an error will be triggered in VBA.

Syntax

```vba
Expression.SaveAs(FileName)
```
Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName (String)</td>
<td>The file name under which the picture is to be saved.</td>
</tr>
</tbody>
</table>

Example:

In the following example the active picture will be saved under the name "Test2.PDL":

```vba
Sub SaveDocumentAs()
 'VBA186
 ActiveDocument.SaveAs("Test2.PDL")
 End Sub
```

See also

LockedByCreatorID Property (Page 2314)
ActiveDocument Property (Page 2107)
Document Object (Page 1949)
VBA Reference (Page 1745)

SaveDefaultConfig Method

Description

Saves the default settings for objects to a PDD file. The file is saved to the "GraCS" folder of the current project.

Syntax

```vba
Expression.SaveDefaultConfig(FileName)
```

Expression

Necessary. An expression or element which returns an object of the "Application" type.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName (String)</td>
<td>The name of the PDD file.</td>
</tr>
</tbody>
</table>

Example:

In the following example the default settings for objects are saved to the file "Test.PDD".

```vba
Sub SaveDefaultConfig()
'VBA187
Application.SaveDefaultConfig ("Test.PDD")
End Sub
```

See also

- Application Object (Page 1909)
- LoadDefaultConfig Method (Page 1863)
- VBA Reference (Page 1745)

SelectAll Method

Description

Selects all the objects in the specified picture and adds them to the selection listing.

Syntax

```
Expression.SelectAll()
```

**Expression**

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--
Example:

In the following example three objects will be inserted in the active picture and then selected.

```vba
Sub SelectAllObjectsInActiveDocument()
  'VBA188
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Dim objEllipse As HMIEllipse
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
  Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
  With objCircle
    .Top = 30
    .Left = 0
    .Height = 15
  End With
  With objRectangle
    .Top = 80
    .Left = 42
    .Width = 25
    .Height = 40
  End With
  With objEllipse
    .Top = 48
    .Left = 162
    .Width = 40
    .Height = 120
    .BackColor = RGB(255, 0, 0)
  End With
  ActiveDocument.Selection.SelectAll
End Sub
```

See also

- SelectedObjects object (Listing) (Page 2060)
- VBA Reference (Page 1745)

SendToBack Method

Description

Sends the selected objects right to the back within their current layer.

Note

If the "SendToBack" method is used, the sequence of HMI objects can change in the HMIObjects listing.
Syntax

Expression.SendToBack()

Expression

Necessary. An expression or element which returns an object of the "Selection" type.

Parameters

--

Example:

In the following example two objects are inserted in the active picture. The object inserted first is then sent to the back:

Sub SendObjectToBack()
'VBA197
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
  .Top = 40
  .Left = 40
  .Selected = False
End With
With objRectangle
  .Top = 40
  .Left = 40
  .Width = 100
  .Height = 50
  .BackColor = RGB(255, 0, 255)
  .Selected = True
End With
MsgBox "The objects circle and rectangle are created" & vbCrLf & "Only the rectangle is selected!"
ActiveDocument.Selection.SendToBack
MsgBox "The selection is moved to the back."
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
VBA Reference (Page 1745)
SetCSLayerVisible Method

Description

Shows or hides the specified CS layer.

Syntax

Expression.SetCSLayerVisible(Index, Val)

Expression

Necessary. An expression or element which returns an object of the "View" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>Defines the CS layer that is going to be shown or hidden. Value range from 1 up to 32.</td>
</tr>
<tr>
<td>Val (Boolean)</td>
<td>TRUE if the specified CS layer is intended to be visible.</td>
</tr>
</tbody>
</table>

Example:

In the following example the second CS layer in the copy of the active picture is hidden (i.e. made invisible):

Sub SetCSLayerVisible()
  'VBA189
  Dim objView As HMIView
  Set objView = ActiveDocument.Views.Add
  objView.Activate
  objView.SetCSLayerVisible 2, False
End Sub

See also

Document Object (Page 1949)
VBA Reference (Page 1745)
Editing Layers with VBA (Page 1657)
SetOpenContext method

Description

The SetOpenContext method sets the password. Password-protected process pictures can then be opened.

Syntax

Expression.SetOpenContext (Password)

Expression

Required. An expression or element which returns an object of the "Documents" type.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password (String)</td>
<td>Password of the available picture.</td>
</tr>
</tbody>
</table>

Example

Several pictures ("A.pdl", "B.pdl" und "C.pdl") are opened in the following example using the same password string "Test123". Enter the password for the pictures to open these. Terminate the SetOpenContext method with an empty string "" to prevent further access to the password.

```vba
Sub OpenProtectedPicture()
    'VBA853
    Documents.SetOpenContext ("Test123")
    Documents.Open ("A.pdl")
    Documents.Open ("B.pdl")
    Documents.Open ("C.pdl")
    Documents.SetOpenContext ("")
End Sub
```

SetDeclutterObjectSize Method

Description

Specifies the size area for fading in and out of objects in the specified picture. If height and width of the object are outside the specified size area, the objects are faded out.

The "ObjectSizeDecluttering" property must be set to TRUE.
### Syntax

Expression.SetDeclutterObjectSize(Min, Max)

### Expression

Necessary. An expression or element which returns an object of the "Document" type.

### Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min (Long)</td>
<td>Lower size range in pixels.</td>
</tr>
<tr>
<td>Max (Long)</td>
<td>Upper size range in pixels.</td>
</tr>
</tbody>
</table>

### Example:

In the following example the settings for the lowest layer are configured in the active picture:

```vba
Sub ConfigureSettingsOfLayer()
    'VBA190
    Dim objLayer As HMILayer
    Set objLayer = ActiveDocument.Layers(1)
    With objLayer
        .MinZoom = 10
        .MaxZoom = 100
        .Name = "Configured with VBA"
    End With
    'Define decluttering of objects:
    With ActiveDocument
        .LayerDecluttering = True
        .ObjectSizeDecluttering = True
        .SetDeclutterObjectSize 50, 100
    End With
End Sub
```

### See also

- ObjectSizeDecluttering Property (Page 2350)
- Document Object (Page 1949)
- VBA Reference (Page 1745)

### SetRTLayerVisible Method

**Description**

Shows or hides the specified RT layer.
syntax

```
Expression.SetRTLayerVisible(Index, Val)
```

Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index (Variant)</td>
<td>Defines the RT layer that is going to be shown or hidden. Value range from 1 to 32.</td>
</tr>
<tr>
<td>Val (Boolean)</td>
<td>TRUE if the specified RT layer is intended to be visible.</td>
</tr>
</tbody>
</table>

Example:

In the following example the first RT layer in the active picture will be made visible:

```
Sub SetRTLayerVisibleWithVBA()
  'VBA191
  ActiveDocument.SetRTLayerVisible 1, False
End Sub
```

See also

- Document Object (Page 1949)
- VBA Reference (Page 1745)
- Editing Layers with VBA (Page 1657)

ShowPropertiesDialog Method

Description

Opens the "Object Properties" dialog.

syntax

```
Expression.ShowPropertiesDialog()
```

Expression

Necessary. An expression or element which returns an object of the "Application" type.
Parameters

```
--
```

Example:

In the following example the "Object Properties" dialog is opened:

```
Sub ShowPropertiesDialog()
  'VBA192
  Application.ShowPropertiesDialog
End Sub
```

See also

- Application Object (Page 1909)
- VBA Reference (Page 1745)

ShowSymbolLibraryDialog Method

Description

Opens the Components Library.

Syntax

```
Expression.ShowSymbolLibraryDialog()
```

Expression

Necessary. An expression or element which returns an object of the "Application" type.

Parameters

```
--
```

Example:

In the following example the Components Library is opened:

```
Sub ShowSymbolLibraryDialog()
  'VBA193
  Application.ShowSymbolLibraryDialog
End Sub
```
See also
Application Object (Page 1909)
VBA Reference (Page 1745)

ShowTagDialog Method

Description
Opens the "Tags" dialog.

Syntax
Expression.ShowTagDialog()

Expression
Necessary. An expression or element which returns an object of the "Application" type.

Parameters
--

Example:
In the following example the "Tags" dialog is opened:

Sub ShowTagDialog()
  'VBA194
  Application.ShowTagDialog
End Sub

See also
Application Object (Page 1909)
VBA Reference (Page 1745)

TileWindowsHorizontally Method

Description
Arranges all open pictures in the Graphics Designer so that they are tiled horizontally.
syntax

Expression.Methode()

Expression

Necessary. An expression or element which returns an object of the "Application" type.

Parameters

--

Example:

In the following example all open pictures in the Graphics Designer are tiled horizontally. For this example to work, you must have opened a number of pictures in the Graphics Designer:

Sub TileWindowsHorizontally()
'VBA195
Application.TileWindowsHorizontally
End Sub

See also

Application Object (Page 1909)

VBA Reference (Page 1745)

TileWindowsVertically Method

Description

Arranges all open pictures in the Graphics Designer so that they are tiled vertically.

syntax

Expression.Methode()

Expression

Necessary. An expression or element which returns an object of the "Application" type.

Parameters

--
Example:

In the following example all open pictures in the Graphics Designer are tiled vertically. For this example to work, you must have opened a number of pictures in the Graphics Designer:

```vba
Sub TileWindowsVertically()
    'VBA196
    Application.TileWindowsVertically
End Sub
```

See also

Application Object (Page 1909)

VBA Reference (Page 1745)

TransformDisplayCoordinate method

Description

Is used internally for PowerCC.

TransformPixelCoordinate method

Description

Is used internally for PowerCC.

Ungroup Method

Description

Ungroups a group object. The objects remain intact.

Syntax

```
Expression.Ungroup(Parameter)
```

Expression

Necessary. An expression or element which returns an object of the "Group" type.

Parameters

---
Example:

In the following example three objects are created in the current picture and a group object is then created from them: The group object is then moved and ungrouped.

Sub DissolveGroup()
'VBA199
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
  .Top = 30
  .Left = 0
  .Selected = True
End With
With objRectangle
  .Top = 80
  .Left = 42
  .Selected = True
End With
With objEllipse
  .Top = 48
  .Left = 162
  .Width = 40
  .BackColor = RGB(255, 0, 0)
  .Selected = True
End With
MsgBox "Objects selected!"
Set objGroup = ActiveDocument.Selection.CreateGroup
MsgBox "Group-object is created."
With objGroup
  .Left = 120
  .Top = 300
MsgBox "Group-object is moved."
.UnGroup
MsgBox "Group is dissolved."
End With
End Sub

See also

Group Object (Page 1978)
CreateGroup Method (Page 1829)
VBA Reference (Page 1745)
Group Objects (Page 1674)
3.6.1.7 Objects and Lists

0-9, A-C

3DBarGraph Object

Description

Represents the "3D Bar" object. The 3DBarGraph object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

VBA Object Name

HMI3DBarGraph

Application

Use the Add method to create a new "3D Bar" object in a picture:

```vba
Sub Add3DBarGraph()
'VBA200
Dim obj3DBarGraph As HMI3DBarGraph
Set obj3DBarGraph = ActiveDocument.HMIObjects.AddHMIObject("3DBar", "HMI3DBarGraph")
End Sub
```

Use "HMIObjects(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

```vba
Sub Edit3DBarGraph()
'VBA201
Dim obj3DBarGraph As HMI3DBarGraph
Set obj3DBarGraph = ActiveDocument.HMIObjects("3DBar")
obj3DBarGraph.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection(Index)" to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA202
    'Select all objects in the picture:
    ActiveDocument.Selection>SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

Object properties

The 3D Bar object possesses the following properties:

- AngleAlpha
- AngleBeta
- Application
- Axe
- BackColor
- Background
- BarDepth
- BarHeight
- BarWidth
- BaseX
- BaseY
- BorderColor
- BorderStyle
- BorderWidth
- Direction
- FillColor
- FillStyle
- GlobalColorScheme
- GlobalShadow
- GroupParent
- Height
- Layer
- Layer00Checked ... Layer10Checked
- Layer00Color ... Layer10Color
- Layer00FillColor ... Layer10FillColor
- Layer0FillStyle ... Layer10FillStyle
- Layer0Value ... Layer10Value
- Left
- LightEffect
- Max.
- Min.
- ObjectName
- Operation
- Parent
- PasswordLevel
- PredefinedAngles
- Process
- Selected
- ShowBadTagState
- TabOrderAlpha
- TabOrderSwitch
- ToolTipText
- Top
- Transparency
- Type
- Visible
- Width
- ZeroPointValue

**See also**

SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
ZeroPointValue Property (Page 2559)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
Process Property (Page 2383)
PredefinedAngels Property (Page 2381)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Name Property (Page 2343)
Min Property (Page 2340)
Max Property (Page 2322)
LightEffect Property (Page 2306)
Left Property (Page 2305)
Layer Property (Page 2293)
Layer00..10Value property (Page 2296)
Height Property (Page 2269)
Direction Property (Page 2213)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderColor Property (Page 2153)
BaseY Property (Page 2147)
BaseX Property (Page 2146)
BarWidth Property (Page 2143)
BarHeight Property (Page 2142)
BarDepth Property (Page 2141)
Background Property (Page 2140)
Axe Property (Page 2129)
AngleBeta Property (Page 2120)
AngleAlpha Property (Page 2119)
Layer00..10Checked property (Page 2293)
Layer00..10Color property (Page 2294)
Application Property (Page 2121)
BackColor Property (Page 2131)
FillColor Property (Page 2231)
FillStyle Property (Page 2235)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
Actions Object (Listing)

Description

Displays a listing of the actions that are configured on an event.

VBA Object Name

HMIActions
Sub CreateVBActionButtonToClickedEvent()
'VBA203
Dim objButton As HMIButton
Dim objCircle As HMICircle
Dim objVBScript As HMIScriptInfo
Dim strVBCode As String
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VB", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircle
    .Top = 100
    .Left = 100
    .BackColor = RGB(255, 0, 0)
End With
With objButton
    .Top = 10
    .Left = 10
    .Text = "Increase Radius"
End With
'define event and assign sourcecode to it:
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
strVBCode = "Dim myCircle" & vbCrLf & "Set myCircle = "
strVBCode = strVBCode & "HMIRuntime.ActiveScreen.ScreenItems(""Circle_VB"")"
strVBCode = strVBCode & vbCrLf & "myCircle.Radius = myCircle.Radius + 5"
With objVBScript
    .SourceCode = strVBCode
End With
End Sub

See also

AddAction Method (Page 1794)
Configuring Event-Driven Actions with VBA (Page 1699)
Parent Property (Page 2359)
Item Property (Page 2283)
Count Property (Page 2201)
Application Property (Page 2121)
ActiveXControl Object

Description

Represents the ActiveX Control object. The ActiveXControl object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

VBA Object Name

HMIActiveXControl

Usage

Use the AddActiveXControl method to insert an ActiveX Control in a picture, for instance. In the following example the ActiveX Control "WinCC Gauge Control" is inserted in the active picture.

```vba
Sub AddActiveXControl()
'VBA204
Dim objActiveXControl As HMIActiveXControl
Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge", "XGAUGE.XGaugeCtrl.1")
With ActiveDocument
  .HMIobjects("WinCC_Gauge").Top = 40
  .HMIobjects("WinCC_Gauge").Left = 40
End With
End Sub
```

See also

- ServerName Property (Page 2415)
- AddActiveXControl Method (Page 1795)
- VBA Reference (Page 1745)
- ActiveX controls (Page 1671)
AdvancedAnalogDisplay object

Description

Represents the "Analog Display (Advanced)" object. The "AdvancedAnalogDisplay" object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.
VBA Object Name

   HMIAdvancedAnalogDisplay

Application

   Use the AddHMIObject method to create a new "Analog Display (Advanced)" object in a picture:

Sub AddAdvancedAnalogDisplay()
'VBA857
Dim objAdvancedAnalogDisplay As HMIAdvancedAnalogDisplay
Set objAdvancedAnalogDisplay = ActiveDocument.HMIObjects.AddHMIObject("Analogdisplay1", "HMIAdvancedAnalogDisplay")
End Sub

   Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

Sub EditAdvancedAnalogDisplay()
'VBA858
Dim objAdvancedAnalogDisplay As HMIAdvancedAnalogDisplay
Set objAdvancedAnalogDisplay = ActiveDocument.HMIObjects("Analogdisplay1")
objAdvancedAnalogDisplay.BackColor_Simulation = RGB(255, 0, 0)
End Sub

See also

   AlarmGoneVisible property (Page 2114)
   AlignmentLeft Property (Page 2117)
   AlignmentTop Property (Page 2118)
   Application Property (Page 2121)
   BackColor Property (Page 2131)
   BackColor_Alarm..._Warning property (Page 2134)
   BackFillColor property (Page 2136)
   BackFillColor_OK property (Page 2136)
   BackFillColor_Simulation property (Page 2137)
   BackFillStyle property (Page 2137)
   BackFillStyle_OK property (Page 2137)
   BackFillStyle_Simulation property (Page 2137)
   BorderColor Property (Page 2153)
   BorderWidth Property (Page 2161)
   UseGlobalAlarmClasses property (Page 2465)
CBackColorOff..ColorOn property (Page 2169)
CBackFlash property (Page 2170)
CollectValue property (Page 2182)
CornerRadius property (Page 2201)
CQBackColorOff..ColorOn property (Page 2203)
CQBackFlash property (Page 2203)
CQTextColorOff..ColorOn property (Page 2203)
CTextColorOff..ColorOn property (Page 2204)
CQTextFlash property (Page 2204)
CTextFlash property (Page 2204)
EventQuitMask property (Page 2222)
EnableFlashing property (Page 2221)
Events Property (Page 2223)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontUnderline Property (Page 2259)
ForeColor Property (Page 2260)
Format property (Page 2263)
ForeColor_Alarm.._Warning property (Page 2261)
GlobalShadow property (Page 2264)
GNQBackColorOff..ColorOn property (Page 2264)
GNQBackFlash property (Page 2265)
GNQTextColorOff..ColorOn property (Page 2265)
GNQTextFlash property (Page 2265)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Layer Property (Page 2293)
LDTooltipTexts Property (Page 2303)
Left Property (Page 2305)
MessageClass Property (Page 2339)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Orientation Property (Page 2356)
OutputValue property (Page 2359)
PaintColor_QualityCodeBad property (Page 2359)
PaintColor_QualityCodeUnCertain property (Page 2359)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
PrioAlarm..Warning property (Page 2383)
PrioBit16..31 property (Page 2383)
Properties Property (Page 2388)
Relevant Property (Page 2398)
Selected Property (Page 2412)
ServerName Property (Page 2415)
ShowBadTagState property (Page 2418)
Simulation property (Page 2419)
SimulationBit property (Page 2419)
Tag property (Page 2435)
tagname property (Page 2436)
tagtype property (Page 2438)
trend property (Page 2448)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
ToolTipText Property (Page 2445)
Top Property (Page 2446)
Transparency property (Page 2447)
Type Property (Page 2450)
UseValueText property (Page 2469)
Visible Property (Page 2551)
Width Property (Page 2553)
ConnectionPoints property (Page 2198)
FlashState property (Page 2252)
ConnectorObjects property (Page 2199)
AdvancedStateDisplay object

Description

Represents the "State Display (Advanced)" object. The "AdvancedStateDisplay" object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIAdvancedStateDisplay

Application

Use the AddHMIObject method to create a new "State Display (Advanced)" object in a picture:

Sub AddAdvancedStateDisplay()
  'VBA859
  Dim objAdvancedStateDisplay As HMIAdvancedStateDisplay
  Set objAdvancedStateDisplay = ActiveDocument.HMIObjects.AddHMIObject("Statedisplay1", "HMIAdvancedStateDisplay")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

Sub EditAdvancedStateDisplay()
  'VBA860
  Dim objAdvancedStateDisplay As HMIAdvancedStateDisplay
  Set objAdvancedStateDisplay = ActiveDocument.HMIObjects("Statedisplay1")
  objAdvancedStateDisplay.PaintColor_QualityCodeBad = RGB(255, 0, 0)
End Sub

See also

UseGlobalAlarmClasses property (Page 2465)
EventQuitMask property (Page 2222)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Tag property (Page 2435)
tagname property (Page 2436)
tagtype property (Page 2438)
ToolTipText Property (Page 2445)
Top Property (Page 2446)
Transparency property (Page 2447)
trend property (Page 2448)
Type Property (Page 2450)
UseEventState property (Page 2464)
Visible Property (Page 2551)
Width Property (Page 2553)
Selected Property (Page 2412)
ServerName Property (Page 2415)
ShowBadTagState property (Page 2418)
Properties Property (Page 2388)
Relevant Property (Page 2398)
Process property (Page 2384)
Process1 property (Page 2384)
Process2 property (Page 2385)
Process3 property (Page 2385)
PaintColor_QualityCodeBad property (Page 2359)
PaintColor_QualityCodeUnCertain property (Page 2359)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
PrioAlarm..Warning property (Page 2383)
PrioBit16..31 property (Page 2383)
MaxIndex property (Page 2322)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Layer Property (Page 2293)
LDTooltipTexts Property (Page 2303)
Left Property (Page 2305)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Index Property (Page 2276)
Events Property (Page 2223)
CollectValue property (Page 2182)
AnalogResultInfo Object

Description

Displays an analog value range and associated property value in the Dynamic dialog. The AnalogResultInfo object is an element of the AnalogResultInfos listing:

VBA Object Name

HMIAnalogResultInfo

Usage

Use the AnalogResultInfo object to return an individual value range and property value. For a detailed example, please refer to "AnalogResultInfos Object (Listing)" in this documentation.

See also

AnalogResultInfos Object (Listing) (Page 1908)
Delete Method (Page 1831)
Value Property (Page 2469)
RangeTo Property (Page 2396)
Parent Property (Page 2359)
Application Property (Page 2121)
AnalogResultInfos Object (Listing)

Description

A listing of AnalogResultInfo objects that contain all the analog value ranges and the associated
property value in the Dynamic dialog.

VBA Object Name

HMIAnalogResultInfos

Usage

Use the Add method to add a new value range in the Dynamic dialog. In the following example
the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will
be assigned and three analog value ranges will be created:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA206
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.Add 50, 40
  .AnalogResultInfos.Add 100, 80
  .AnalogResultInfos.ElseCase = 100
End With
End Sub
```
Use AnalogResultInfos to return the AnalogResultInfos listing. In this example the value ranges created in the above example will be output:

```vba
Sub ShowAnalogResultInfosOfCircleRadius()
'VBA207
Dim colAResultInfos As HMIAnalogResultInfos
Dim objAResultInfo As HMIAnalogResultInfo
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Dim iAnswer As Integer
Dim varRange As Variant
Dim varValue As Variant
Set objCircle = ActiveDocument.HMIObjects("Circle_A")
Set objDynDialog = objCircle.Radius.Dynamic
Set colAResultInfos = objDynDialog.AnalogResultInfos
For Each objAResultInfo In colAResultInfos
    varRange = objAResultInfo.RangeTo
    varValue = objAResultInfo.value
    iAnswer = MsgBox("Ranges of values from Circle_A-Radius:" & vbCrLf & "Range of value to: " & varRange & vbCrLf & "Value of property: " & varValue, vbOKCancel)
    If vbCancel = iAnswer Then Exit For
Next objAResultInfo
End Sub
```

See also

Add Method (AnalogResultInfos Listing) (Page 1787)
Parent Property (Page 2359)
Item Property (Page 2283)
ElseCase Property (Page 2220)
Count Property (Page 2201)
Application Property (Page 2121)

Application Object

Description

Represents the Graphics Designer editor. The Application object contains properties and methods that return objects from the top layer. For example ActiveDocument returns a Document object.
VBA Object Name

HMIApplication

Usage

Use Application to return the Application object. In the following example the application version is output:

Sub ShowApplicationVersion()
    'VBA208
    MsgBox Application.Version
End Sub

See also

ShowTagDialog Method (Page 1890)
CurrentDesktopLanguage Property (Page 2205)
TileWindowsVertically Method (Page 1891)
TileWindowsHorizontally Method (Page 1890)
ShowSymbolLibraryDialog Method (Page 1889)
ShowPropertiesDialog Method (Page 1888)
SaveDefaultConfig Method (Page 1881)
LoadDefaultConfig Method (Page 1863)
CascadeWindows Method (Page 1814)
ArrangeMinimizedWindows Method (Page 1811)
Activate Method (Page 1785)
VBA Reference (Page 1745)
WindowState Property (Page 2557)
Visible Property (Page 2551)
Version Property (Page 2550)
VBE Property (Page 2550)
VBAVersion Property (Page 2549)
SymbolLibraries Property (Page 2428)
ProjectType Property (Page 2387)
ProfileName Property (Page 2385)
Parent Property (Page 2359)
ApplicationWindow Object

Description

Multiple Objects

HMIDefaultObjects

HMIObjects

Selection

ApplicationWindow

Represents the "Application Window" object. The ApplicationWindow object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.
VBA Object Name

HMIApplicationWindow

Usage

Use the Add method to create a new "Application Window" object in a picture:

Sub AddApplicationWindow()
'VBA209
Dim objApplicationWindow As HMIApplicationWindow
Set objApplicationWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow", "HMIApplicationWindow")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditApplicationWindow()
'VBA210
Dim objApplicationWindow As HMIApplicationWindow
Set objApplicationWindow = ActiveDocument.HMIObjects("AppWindow")
objApplicationWindow.Sizeable = True
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA211
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

Caption Property (Page 2168)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
WindowBorder Property (Page 2555)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
Sizeable Property (Page 2421)
OnTop Property (Page 2352)
Name Property (Page 2343)
Moveable Property (Page 2342)
MaximizeButton Property (Page 2322)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
CloseButton Property (Page 2182)
Application Property (Page 2121)
Events Property (Page 2223)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
ToolTipText Property (Page 2445)
ConnectionPoints property (Page 2198)
Template property (Page 2438)
ConnectorObjects property (Page 2199)
BarGraph Object

Description

Represents the "Bar" object. The BarGraph object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default settings of property values of all Standard, Windows and Smart objects.

VBA Object Name

HMIBarGraph

Application

Use the Add method to create a new "Bar" object in a picture:

```vba
Sub AddBarGraph()
'VBA212
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

```vba
Sub EditBarGraph()
'VBA213
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects("Bar1")
objBarGraph.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection(Index)" to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA214
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

TypeWarningHigh Property (Page 2458)
Max Property (Page 2322)
FillColor Property (Page 2231)
BorderStyle Property (Page 2160)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
ZeroPointValue Property (Page 2559)
ZeroPoint Property (Page 2558)
Width Property (Page 2553)
 WarningLow Property (Page 2553)
WarningHigh Property (Page 2552)
Visible Property (Page 2551)
TypeWarningLow Property (Page 2459)
TypeToleranceLow Property (Page 2457)
TypeToleranceHigh Property (Page 2456)
TypeLimitLow4 Property (Page 2454)
TypeLimitHigh4 Property (Page 2453)
TypeAlarmLow Property (Page 2452)
TypeAlarmHigh Property (Page 2451)
Trend Property (Page 2448)
TrendColor Property (Page 2449)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
ToleranceLow Property (Page 2442)
ToleranceHigh Property (Page 2442)
ScalingType Property (Page 2406)
Scaling Property (Page 2405)
ScaleTicks Property (Page 2404)
ScaleColor Property (Page 2403)
RightComma Property (Page 2399)
Process Property (Page 2383)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Name Property (Page 2343)
Min Property (Page 2340)
Marker Property (Page 2321)
LongStrokesTextEach Property (Page 2319)
LongStrokesSize Property (Page 2319)
LongStrokesOnly Property (Page 2318)
LongStrokesBold Property (Page 2318)
LimitLow4 Property (Page 2308)
LimitHigh4 Property (Page 2307)
Left Property (Page 2305)
LeftComma Property (Page 2305)
Layer Property (Page 2293)
HysteresisRange Property (Page 2274)
Hysteresis Property (Page 2273)
Height Property (Page 2269)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontBold Property (Page 2256)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
Exponent Property (Page 2227)
Direction Property (Page 2213)
ColorWarningLow Property (Page 2193)
ColorWarningHigh Property (Page 2192)
ColorToleranceLow Property (Page 2190)
ColorToleranceHigh Property (Page 2189)
ColorLimitLow4 Property (Page 2187)
ColorLimitHigh4 Property (Page 2186)
ColorChangeType Property (Page 2185)
ColorAlarmLow Property (Page 2183)
ColorAlarmHigh Property (Page 2182)
CheckWarningLow Property (Page 2179)
CheckWarningHigh Property (Page 2178)
CheckToleranceLow Property (Page 2177)
CheckToleranceHigh Property (Page 2176)
CheckLimitLow4 Property (Page 2174)
CheckLimitHigh4 Property (Page 2172)
CheckAlarmLow Property (Page 2171)
CheckAlarmHigh Property (Page 2170)
BorderWidth Property (Page 2161)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
BackColor3 Property (Page 2133)
BackColor2 Property (Page 2133)
AxisSection Property (Page 2130)
Average Property (Page 2128)
Alignment Property (Page 2116)
AlarmLow Property (Page 2115)
AlarmHigh Property (Page 2114)
FillStyle2 Property (Page 2237)
BinaryResultInfo Object

Description

```
DynamicDialogInfo
  BinaryResultInfo
```

Displays both the binary (boolean) value ranges and the associated property values in the Dynamic dialog.

VBA Object Name

HMIBinaryResultInfo
Usage

Use BinaryResultInfo to return the BinaryResultInfo object. In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and the associated property values will be assigned to both the binary value ranges:

Sub AddDynamicDialogToCircleRadiusTypeBinary()
'VBA215
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_C", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeBool
  .BinaryResultInfo.NegativeValue = 20
  .BinaryResultInfo.PositiveValue = 40
End With
End Sub

See also

VBA Reference (Page 1745)
PositiveValue Property (Page 2380)
Parent Property (Page 2359)
NegativeValue Property (Page 2344)
Application Property (Page 2121)

BitResultInfo Object

Description

Displays both the value ranges for bit set/not set and the associated property values in the Dynamic dialog.

VBA Object Name

HMIBitResultInfo
Usage

Use BitResultInfo to return a BitResultInfo object. In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned, the bit to be set will be defined and the associated property values will be assigned to the “set”/”not set” states:

Sub AddDynamicDialogToCircleRadiusTypeBit()
'VBA216
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_B", "HMICircle")
'Tag "NewDynamic1" must exist
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'"
With objDynDialog
  .ResultType = hmiResultTypeBit
  .BitResultInfo.BitNumber = 1
  .BitResultInfo.BitSetValue = 40
  .BitResultInfo.BitNotSetValue = 80
End With
End Sub

See also

Delete Method (Page 1831)
VBA Reference (Page 1745)
BitSetValue Property (Page 2150)
BitNumber Property (Page 2148)
BitNotSetValue Property (Page 2148)
Application Property (Page 2121)
Parent Property (Page 2359)
**Button Object**

**Description**

- Represents the "Button" object. The Button object is an element of the following listings:
  - HMIObjects: Contains all objects of a picture.
  - Selection: Contains all selected objects of a picture.
  - HMIDefaultObjects: Contains the default settings of property values of all Standard, Windows and Smart objects.

**VBA Object Name**

HMIButton

**Application**

Use the Add method to create a new "Button" object in a picture:

```vba
Sub AddButton()
    'VBA217
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button", "HMIButton")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

```vba
Sub EditButton()
    'VBA218
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects("Button")
    objButton.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection(Index)" to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA219
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ForeFlashColorOn Property (Page 2262)
BorderColorBottom Property (Page 2155)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
TooltipText Property (Page 2445)
Text Property (Page 2439)
PictureUp Property (Page 2374)
PictureDown Property (Page 2372)
PasswordLevel Property (Page 2362)
Orientation Property (Page 2356)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Hotkey Property (Page 2272)
Height Property (Page 2269)
ForeFlashColorOff Property (Page 2261)
ForeColor Property (Page 2260)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FontBold Property (Page 2256)
FlashRateForeColor Property (Page 2251)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashForeColor Property (Page 2242)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
DisplayOptions Property (Page 2215)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColorTop Property (Page 2155)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
BackBorderWidth Property (Page 2131)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
AdaptBorder Property (Page 2111)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDFonts Property (Page 2298)
LDTexsts Property (Page 2303)
CheckBox Object

Description
Represents the "Check Box" object. The CheckBox object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default settings of property values of all Standard, Windows and Smart objects.

**VBA Object Name**

HMICheckBox

**Application**

Use the Add method to create a new "Check Box" object in a picture:

```vba
Sub AddCheckBox()
    'VBA220
    Dim objCheckBox As HMICheckBox
    Set objCheckBox = ActiveDocument.HMIObjects.AddHMIObject("CheckBox", "HMICheckBox")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

```vba
Sub EditCheckBox()
    'VBA221
    Dim objCheckBox As HMICheckBox
    Set objCheckBox = ActiveDocument.HMIObjects("CheckBox")
    objCheckBox.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection(Index)" to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
    'VBA222
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

**See also**

- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Application Property (Page 2121)
AdaptBorder Property (Page 2111)
AlignmentLeft Property (Page 2117)
AlignmentTop Property (Page 2118)
BackColor Property (Page 2131)
BackFlashColorOff Property (Page 2138)
BackFlashColorOn Property (Page 2139)
BorderBackColor Property (Page 2152)
BorderColor Property (Page 2153)
BorderFlashColorOff Property (Page 2157)
BorderFlashColorOn Property (Page 2159)
BorderStyle Property (Page 2160)
BorderWidth Property (Page 2161)
BoxAlignment Property (Page 2163)
BoxCount Property (Page 2164)
Events Property (Page 2223)
FillColor Property (Page 2231)
Filling Property (Page 2232)
FillingIndex Property (Page 2233)
FillStyle Property (Page 2235)
FlashBackColor Property (Page 2239)
FlashBorderColor Property (Page 2240)
FlashForeColor Property (Page 2242)
FlashRateBackColor Property (Page 2247)
FlashRateBorderColor Property (Page 2249)
FlashRateForeColor Property (Page 2251)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontUnderline Property (Page 2259)
ForeColor Property (Page 2260)
ForeFlashColorOff Property (Page 2261)
ForeFlashColorOn Property (Page 2262)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Index Property (Page 2276)
Layer Property (Page 2293)
LDFonts Property (Page 2298)
LDTtexts Property (Page 2303)
LDTOoltipTexts Property (Page 2303)
Left Property (Page 2305)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
OperationMessage Property (Page 2354)
Orientation Property (Page 2356)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Process Property (Page 2383)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Text Property (Page 2439)
ToolTipText Property (Page 2445)
Top Property (Page 2446)
Transparency property (Page 2447)
Type Property (Page 2450)
Visible Property (Page 2551)
Width Property (Page 2553)
WindowsStyle property (Page 2557)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
Circle Object

Description

Represents the "Circle" object. The Circle object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIcircle

Usage

Use the Add method to create a new "Circle" object in a picture:

```vba
Sub AddCircle()
  'VBA223
  Dim objCircle As HMIcircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIOntent("Circle", "HMIcircle")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditCircle()
  'VBA224
  Dim objCircle As HMIcircle
  Set objCircle = ActiveDocument.HMIObjects("Circle")
  objCircle.BorderColor = RGB(255, 0, 0)
End Sub
```
Sub ShowNameOfFirstSelectedObject()
  'VBA225
  'Select all objects in the picture:
  ActiveDocument.Selection.SelectAll
  'Get the name from the first object of the selection:
  MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

FillColor Property (Page 2231)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
Radius Property (Page 2394)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTOoltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
CircularArc Object

Description

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMICircularArc

Usage

Use the Add method to create a new "Circular Arc" object in a picture:

```vba
Sub AddCircularArc()
    'VBA226
    Dim objCircularArc As HMICircularArc
    Set objCircularArc = ActiveDocument.HMIObjects.AddHMIObject("CircularArc", "HMICircularArc")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditCircularArc()
    'VBA227
    Dim objCircularArc As HMICircularArc
    Set objCircularArc = ActiveDocument.HMIObjects("CircularArc")
    objCircularArc.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA228
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

- HMIObjec.ts Object (Listing) (Page 1990)
- BorderBackColor Property (Page 2152)
- SelectedObjects object (Listing) (Page 2060)
- HMIDefaultObjects Object (Listing) (Page 1985)
- AddHMIObject Method (Page 1801)
- VBA Reference (Page 1745)
- Editing Objects with VBA (Page 1660)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Top Property (Page 2446)
- ToolTipText Property (Page 2445)
- StartAngle Property (Page 2426)
- Radius Property (Page 2394)
- PasswordLevel Property (Page 2362)
- Operation Property (Page 2353)
- Left Property (Page 2305)
- Layer Property (Page 2293)
- Height Property (Page 2269)
- FlashRateBorderColor Property (Page 2249)
- FlashBorderColor Property (Page 2240)
- EndAngle Property (Page 2222)
- BorderWidth Property (Page 2161)
- BorderStyle Property (Page 2160)
- BorderFlashColorOn Property (Page 2159)
- BorderFlashColorOff Property (Page 2157)
Collection object

Description

Allows access to a listing of objects of the same type, for example, "Documents" objects.

See also

Application Property (Page 2121)
Count Property (Page 2201)
Item Property (Page 2283)
Parent Property (Page 2359)
ComboBox object

Description

Represents the "ComboBox" object. The ComboBox object is an element of the following lists:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all default, smart, window and tube objects.

VBA object name

**HMIComboBox**

Usage

Use the Add method to create a new "ComboBox" object in a picture:

```vba
Sub AddComboBox()
"VBA822
Dim objComboBox As HMIComboBox
Set objComboBox = ActiveDocument.HMIObjects.AddHMIObject("ComboBox", "HMIComboBox")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditComboBox()
"VBA850
Dim objComboBox As HMIComboBox
Set objComboBox = ActiveDocument.HMIObjects("ComboBox")
objComboBox.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA824
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ObjectName Property (Page 2347)
Left Property (Page 2305)
Layer Property (Page 2293)
Top Property (Page 2446)
Width Property (Page 2553)
Height Property (Page 2269)
NumberLines Property (Page 2346)
ForeColor Property (Page 2260)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackColor Property (Page 2131)
BorderStyle Property (Page 2160)
BorderWidth Property (Page 2161)
FillColor Property (Page 2231)
FillStyle Property (Page 2235)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontUnderline Property (Page 2259)
AlignmentLeft Property (Page 2117)
GlobalShadow property (Page 2264)
Index Property (Page 2276)
Text Property (Page 2439)
Operation Property (Page 2353)
ConnectionPoints Object (Listing)

Description

```
Multiple Objects
```

```
HMIObjects
```

```
Selection
```

```
ConnectionPoints
```

The listing returns the number of points to which the connector can be appended in the specified object.

VBA object name

```
HMIConnectionPoints
```
Object properties

The ConnectionPoints object possesses the following properties:

- Application
- Count
- Item
- Parent

Example 1

In this example, a rectangle is inserted and the number of connection points is output:

```
Sub CountConnectionPoints()
    'VBA229
    Dim objRectangle As HMIRectangle
    Dim objConnPoints As HMIConnectionPoints
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    Set objConnPoints = ActiveDocument.HMIObjects("Rectangle1").ConnectionPoints
    MsgBox "Rectangle1 has " & objConnPoints.Count & " connection points."
End Sub
```

Example 2:

In this example, a text field is inserted and the connection points are accessed via "ConnectionPoints.Item". The coordinates of the connection points are shown in an output window.

```
Sub GetConnectionPoints()
    'VBA825
    Dim xPos As Long
    Dim yPos As Long
    Dim objConnPoints As HMIConnectionPoints
    Set objDoc = Application.ActiveDocument
    Set objObject = objDoc.HMIObjects.AddHMIObject("Text","HMIStaticText")
    Set objConnPoints = ActiveDocument.HMIObjects("Text").ConnectionPoints
    For i = 1 To objConnPoints.Count
        xPos = objObject.ConnectionPoints.Item(i)(0)
        yPos = objObject.ConnectionPoints.Item(i)(1)
        MsgBox "Coordinates " & i & ". ConnectionPoint:" & Chr(13) & "x: " & xPos & Chr(13) & "y: " & yPos
    Next
End Sub
```

The diagram below shows the positions of the 4 connection points of the text field.
Note
If you activate the connection points of a connector with VBA, the connection point index begins with "1".

If you determine the connection points in the property window of the connector in the graphical interface, the connection point index begins with "0".

The index numbers e.g. of the lower connection point in the picture are assigned as follows:
- VBA: Index = 3
- Graphical interface: Index = 2

See also
Parent Property (Page 2359)
Item Property (Page 2283)
Count Property (Page 2201)
Application Property (Page 2121)

CustomizedObject Object

Description
Represents the object called "Customized Object". The CustomizedObject object is an element of the following listings:

- **Objects:** Contains all objects of a picture.
- **Selection:** Contains all selected objects of a picture.

In the case of the CustomizedObject object, the only properties that are available in the object are those that you have selected in the "Configuration" dialog for the customized object concerned.

**Note**

You cannot configure the CustomizedObject object with VBA.

Further information regarding customized objects can be found in the WinCC documentation under "Customized Object".

**VBA Object Name**

HMICustomizedObject

**Application**

Use the CreateCustomizedObject Method with the Selection listing to create a new "Customized Object" object in a picture:

```vba
Sub CreateCustomizedObject()
  'VBA230
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Dim objCustomizedObject As HMICustomizedObject
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
  With objCircle
    .Left = 10
    .Top = 10
    .Selected = True
  End With
  With objRectangle
    .Left = 50
    .Top = 50
    .Selected = True
  End With
  MsgBox "objects created and selected!"
  Set objCustomizedObject = ActiveDocument.Selection.CreateCustomizedObject
  objCustomizedObject.ObjectName = "Customer-Object"
End Sub
```
Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name:

Sub EditCustomizedObject()
'VBA231
Dim objCustomizedObject As HMICustomizedObject
Set objCustomizedObject = ActiveDocument.HMIObjects("Customer-Object")
MsgBox objCustomizedObject.ObjectName
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
Destroy Method (Page 1835)
Delete Method (Page 1831)
CreateCustomizedObject Method (Page 1825)
How to Edit a Customized Object with VBA (Page 1683)
VBA Reference (Page 1745)
Customized Objects (Page 1682)
Editing Objects with VBA (Page 1660)
Application Property (Page 2121)
Events Property (Page 2223)
GroupParent Property (Page 2268)
Height Property (Page 2269)
InheritState property (Page 2278)
Layer Property (Page 2293)
LDTooltipTexts Property (Page 2303)
Left Property (Page 2305)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
DataLanguage Object

Description

![Object Diagram]

Represents the installed project language, which is identified by its name and language identifier. The DataLanguage object is an element of the DataLanguages listing:

The list of language codes is available in the WinCC documentation (Index > Language Code). The hexadecimal value specified in the list has to be converted to its equivalent decimal value.

VBA Object Name

HMIDataLanguage

Usage

Use the DataLanguages property to return an individual DataLanguage object. In the following example the first installed project language is output:

```vba
Sub ShowFirstObjectOfCollection()
  'VBA232
  Dim strName As String
  strName = ActiveDocument.Application.AvailableDataLanguages(1).LanguageName
  MsgBox strName
End Sub
```
See also

DataLanguages Object (Listing) (Page 1942)
VBA Reference (Page 1745)
Language-Dependent Configuration with VBA (Page 1624)
Parent Property (Page 2359)
LanguageName Property (Page 2290)
LanguageID Property (Page 2289)
Application Property (Page 2121)

DataLanguages Object (Listing)

Description

A listing of the DataLanguage objects that represent all the installed project languages.

VBA Object Name

HMIDataLanguages
VBA for Automated Configuration
3.6 VBA Reference

Usage
Use the AvailableDataLanguages property to return the DataLanguages listing. In the following
example the installed project language is output:

Sub ShowDataLanguage()
'VBA233
Dim colDataLanguages As HMIDataLanguages
Dim objDataLanguage As HMIDataLanguage
Dim strLanguages As String
Dim iCount As Integer
iCount = 0
Set colDataLanguages = Application.AvailableDataLanguages
For Each objDataLanguage In colDataLanguages
If "" <> strLanguages Then strLanguages = strLanguages & "/"
strLanguages = strLanguages & objDataLanguage.LanguageName & " "
'Every 15 items of datalanguages output in a messagebox
If 0 = iCount Mod 15 And 0 <> iCount Then
MsgBox strLanguages
strLanguages = ""
End If
iCount = iCount + 1
Next objDataLanguage
MsgBox strLanguages
End Sub

See also
Language-Dependent Configuration with VBA (Page 1624)
DataLanguage Object (Page 1941)
Item Method (Page 1859)
VBA Reference (Page 1745)
Parent Property (Page 2359)
Count Property (Page 2201)
Application Property (Page 2121)

DataSetObj object
Description
The "DataSetObj" object serves as a container for the internal storage of data of the user
objects or faceplate types. The DataSetObj object is an element of the following listings:
● Objects: Contains all objects of a picture.
● Selection: Contains all selected objects of a picture.

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA

1943


VBA Object Name

HMIDataSetObj

See also

Application Property (Page 2121)
ConnectionPoints property (Page 2198)
Events Property (Page 2223)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Layer Property (Page 2293)
LDFonts Property (Page 2298)
Left Property (Page 2305)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
ToolTipText Property (Page 2445)
Top Property (Page 2446)
Type Property (Page 2450)
Visible Property (Page 2551)
Width Property (Page 2553)
ConnectorObjects property (Page 2199)
DestLink Object

Description

DirectConnection

DestLink

SourceLink

Represents the destination for a direct connection.

VBA Object Name

HMIDestLink
Usage

Use the DestinationLink property to return the DestLink object. In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
'VBA234
Dim objButton As HMIButton
Dim objRectangleA As HMIRectangle
Dim objRectangleB As HMIRectangle
Dim objEvent As HMIEvent
Dim objDirConnection As HMIDirectConnection
Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objRectangleA
  .Top = 100
  .Left = 100
End With
With objRectangleB
  .Top = 250
  .Left = 400
  .BackColor = RGB(255, 0, 0)
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 90
  .Height = 50
  .Text = "SetPosition"
End With
'
'Directconnection is initiated on mouseclick:
Set objDirConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
With objDirConnection
  'Sourceobject: Property "Top" of "Rectangle_A"
  .SourceLink.Type = hmiSourceTypeProperty
  .SourceLink.ObjectName = "Rectangle_A"
  .SourceLink.AutomationName = "Top"
  ',
  'Targetobject: Property "Left" of "Rectangle_B"
  .DestinationLink.Type = hmiDestTypeProperty
  .DestinationLink.ObjectName = "Rectangle_B"
  .DestinationLink.AutomationName = "Left"
End With
End Sub
See also

- DirectConnection Object (Page 1947)
- VBA Reference (Page 1745)
- Type Property (Page 2450)
- ObjectName Property (Page 2347)
- AutomationName Property (Page 2124)
- Parent Property (Page 2359)

**DirectConnection Object**

**Description**

- **Multiple Objects**
- **DirectConnection**
- **ActionDynamic**

Represents a direct connection.

**VBA Object Name**

HMIDirectConnection
Usage

Use the DestinationLink and SourceLink properties to configure the source and destination of a direct connection. In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

```vba
Sub DirectConnection()
    'VBA235
    Dim objButton As HMIButton
    Dim objRectangleA As HMIRectangle
    Dim objRectangleB As HMIRectangle
    Dim objEvent As HMIEvent
    Dim objDirConnection As HMIDirectConnection
    Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
    Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    With objRectangleA
        .Top = 100
        .Left = 100
    End With
    With objRectangleB
        .Top = 250
        .Left = 400
        .BackColor = RGB(255, 0, 0)
    End With
    With objButton
        .Top = 10
        .Left = 10
        .Width = 90
        .Height = 50
        .Text = "SetPosition"
    End With
    'Directconnection is initiated on mouseclick:
    Set objDirConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
    With objDirConnection
        'Sourceobject: Property "Top" of "Rectangle_A"
        .SourceLink.Type = hmiSourceTypeProperty
        .SourceLink.ObjectName = "Rectangle_A"
        .SourceLink.AutomationName = "Top"
    End With
    'Targetobject: Property "Left" of "Rectangle_B"
    .DestinationLink.Type = hmiDestTypeProperty
    .DestinationLink.ObjectName = "Rectangle_B"
    .DestinationLink.AutomationName = "Left"
End Sub
```
See also

- DestinationLink Property (Page 2211)
- SourceLink Object (Page 2065)
- DestLink Object (Page 1945)
- VBA Reference (Page 1745)
- SourceLink Property (Page 2422)
- Application Property (Page 2121)
- Parent Property (Page 2359)

Document Object

Description

Displays a picture in Graphics Designer. The document object is an element of the documents listing.

VBA Object Name

HMI Document

Usage

Use Documents(Index) to return an individual document object. In the following example the file name of the first picture is displayed:

Sub ShowFirstObjectOfCollection()
'VBA236
Dim strName As String
strName = Application.Documents(3).Name
MsgBox strName
End Sub
You may also use the object "Me" if you wish to address the current document:

```vba
Sub ShowDocumentName()
    'VBA812
    Dim obj As Document
    set obj = Me
    MsgBox obj.Name
End Sub
```

For example, use the SaveAs method to save the picture under a different name. In the following example the first picture will be saved under the name "CopyOfPicture1":

```vba
Sub SaveDocumentAs()
    'VBA237
    Application.Documents(3).SaveAs ("CopyOfPicture1")
End Sub
```

See also

- Editing Pictures with VBA (Page 1654)
- GridHeight Property (Page 2267)
- Documents Object (Listing) (Page 1952)
- SetRTLLayerVisible Method (Page 1887)
- SaveAs Method (Page 1880)
- Save Method (Page 1879)
- PrintProjectDocumentation Method (Page 1870)
- PasteClipboard Method (Page 1868)
- MoveSelection Method (Page 1865)
- IsRTLLayerVisible Method (Page 1858)
- Export Method (Page 1840)
- CopySelection Method (Page 1823)
- Close Method (Page 1818)
- VBA Reference (Page 1745)
- ExtendedZoomingEnable Property (Page 2228)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Views Property (Page 2550)
- UpdateCycle Property (Page 2463)
- TabOrderOtherAction Property (Page 2432)
TabOrderMouse Property (Page 2432)
TabOrderKeyboard Property (Page 2431)
TabOrderAllHMIObjects Property (Page 2430)
SnapToGrid Property (Page 2422)
HMIDefaultObjects Object (Listing) (Page 1985)
Selection Property (Page 2413)
Properties Property (Page 2388)
Path Property (Page 2363)
PasswordLevel Property (Page 2362)
Parent Property (Page 2359)
Operation Property (Page 2353)
Name Property (Page 2343)
LockedByCreatorID Property (Page 2314)
LastChange Property (Page 2292)
HMIObjects Property (Page 2272)
Hide Property (Page 2270)
Height Property (Page 2269)
GridWidth Property (Page 2267)
GridColor Property (Page 2266)
Grid Property (Page 2265)
FillStyle Property (Page 2235)
Events Property (Page 2223)
CustomToolbars Property (Page 2208)
CustomMenues Property (Page 2207)
CursorMode Property (Page 2206)
BackColor Property (Page 2131)
Application Property (Page 2121)
BackPictureAlignment property (Page 2141)
BackPictureName property (Page 2141)
CommonVBSCode Property (Page 2194)
CommonVBSEventArea property (Page 2195)
CommonVBSPROPERTYArea property (Page 2196)
GlobalColorScheme property (Page 2264)
LayerDecluttering Property (Page 2296)
Layers Property (Page 2297)
Modified Property (Page 2341)
ObjectSizeDecluttering Property (Page 2350)
PdlProtection property (Page 2364)
GetDeclutterObjectSize method (Page 1830)
SetDeclutterObjectSize Method (Page 1886)
FireConnectionEvents method (Page 1843)
TransformDisplayCoordinate method (Page 1892)
TransformPixelCoordinate method (Page 1892)
FillBackColor property (Page 2231)

Documents Object (Listing)

Description

VBA Object Name

HMIDocuments

Usage

Note

Use the "ActiveDocument" property if you wish to refer to the active picture.
Use the Documents property to return the Documents listing. In the following example the names of all open pictures are output:

```vba
Sub ShowDocuments()
  Dim colDocuments As Documents
  Dim objDocument As Document
  Set colDocuments = Application.Documents
  For Each objDocument In colDocuments
    MsgBox objDocument.Name
  Next objDocument
End Sub
```

Use the Add method to add a new Document object to the Documents listing. In the following example a new picture is created:

```vba
Sub AddNewDocument()
  Dim objDocument As Document
  Set objDocument = Application.Documents.Add
End Sub
```

See also

- Add Method (Page 1786)
- Document Object (Page 1949)
- SaveAll Method (Page 1880)
- Open Method (Page 1867)
- CloseAll Method (Page 1819)
- Close Method (Page 1818)
- VBA Reference (Page 1745)
- Editing Pictures with VBA (Page 1654)
- Parent Property (Page 2359)
- Count Property (Page 2201)
- Application Property (Page 2121)
- ActiveDocument Property (Page 2107)
- Item Property (Page 2283)
- SetOpenContext method (Page 1886)
- ConvertWM method (Page 1822)
DotNetControl object

Description

Represents the "DotNetControl" object. The DotNetControl object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

VBA Object Name

HMIDotNetControl

Application

Use the AddDotNetControl method to insert a DotNetControl in a picture.

In the following example, the "NETControl" object from the Global Assembly Cache is inserted in the active picture.

'VBA851
Dim DotNetControl As HMIDotNetControl

See also

AddDotNetControl method (Page 1797)
Delete Method (Page 1831)
Application Property (Page 2121)
AssemblyInfo property (Page 2122)
ControlType property (Page 2201)
Events Property (Page 2223)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Layer Property (Page 2293)
LDToltipTexts Property (Page 2303)
Left Property (Page 2305)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
DynamicDialog Object

Description

Represents the Dynamic dialog. You can use the dynamic dialog to make the properties of pictures and objects respond dynamically to different value ranges.
Define the value range with the aid of the ResultType property.

VBA Object Name

HMIDynamicDialog
Usage

Use the DynamicDialog object to make an object property dynamic. In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and three analog value ranges will be created:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA240
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .Trigger.VariableTriggers.Add "NewDynamic2", hmiVariableCycleType_5s
  .AnalogResultInfos.Add 50, 40
  .AnalogResultInfos.Add 100, 80
  .AnalogResultInfos.ElseCase = 100
End With
End Sub

See also

Delete Method (Page 1831)
ConvertToScript Method (Page 1820)
CheckSyntax Method (Page 1817)
VariableStateValues Property (Page 2546)
VariableStateChecked Property (Page 2545)
Trigger Property (Page 2449)
SourceCode Property (Page 2424)
ScriptType Property (Page 2407)
ResultType Property (Page 2399)
Parent Property (Page 2359)
Compiled Property (Page 2197)
BitResultInfo Property (Page 2149)
BinaryResultInfo Property (Page 2147)
Application Property (Page 2121)
AnalogResultInfos Property (Page 2118)
Prototype Property (Page 2389)
QualityCodeStateChecked Properties (Page 2390)
QualityCodeStateValues Property (Page 2392)
**Ellipse Object**

**Description**

Represents the "Ellipse" object. The Ellipse object is an element of the following listings:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

HMIEllipse

**Usage**

Use the Add method to create a new "Ellipse" object in a picture:

```vba
Sub AddEllipse()
  'VBA241
  Dim objEllipse As HMIEllipse
  Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("Ellipse", "HMIEllipse")
End Sub
```
Use "HMIObjects"(Index) to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditEllipse()
'VBA242
Dim objEllipse As HMIEllipse
Set objEllipse = ActiveDocument.HMIObjects("Ellipse")
objEllipse.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing.

Sub ShowNameOfFirstSelectedObject()
'VBA243
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

FillingIndex Property (Page 2233)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
RadiusWidth Property (Page 2395)
RadiusHeight Property (Page 2394)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
EllipseArc Object

Description

Represents the "Ellipse Arc" object. The EllipseArc object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIEllipseArc

Usage

Use the Add method to create a new "Ellipse Arc" object in a picture:

Sub AddEllipseArc()
  'VBA244
  Dim objEllipseArc As HMIEllipseArc
  Set objEllipseArc = ActiveDocument.HMIObjects.AddHMIObject("EllipseArc", "HMIEllipseArc")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditEllipseArc()
  'VBA245
  Dim objEllipseArc As HMIEllipseArc
  Set objEllipseArc = ActiveDocument.HMIObjects("EllipseArc")
  objEllipseArc.BorderColor = RGB(255, 0, 0)
End Sub
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
  'VBA246
  'Select all objects in the picture:
  ActiveDocument.Selection.SelectAll
  'Get the name from the first object of the selection:
  MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

Use the "HMIDefaultObjects(Index)" to return an object from the HMIDefaultObjects Listing:

Sub EditDefaultPropertiesOfEllipseArc()
  'VBA247
  Dim objEllipseArc As HMIEllipseArc
  Set objEllipseArc = Application.DefaultHMIObjects("HMIEllipseArc")
  objEllipseArc.BorderColor = RGB(255, 255, 0)
  'create new "EllipseArc"-object
  Set objEllipseArc = ActiveDocument.HMIObjects.AddHMIObject("EllipseArc2", "HMIEllipseArc")
End Sub

See also

ToolTipText Property (Page 2445)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
StartAngle Property (Page 2426)
RadiusWidth Property (Page 2395)
RadiusHeight Property (Page 2394)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashBorderColor Property (Page 2240)
EndAngle Property (Page 2222)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
**EllipseSegment Object**

**Description**

 Represents the "Ellipse Segment" object. The EllipseSegment object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

HMIEllipseSegment

**Usage**

Use the Add method to create a new "Ellipse Segment" object in a picture:

```vba
Sub AddEllipseSegment()
    'VBA248
    Dim objEllipseSegment As HMIEllipseSegment
    Set objEllipseSegment = ActiveDocument.HMIObjects.AddHMIObject("EllipseSegment", "HMIEllipseSegment")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditEllipseSegment()
    'VBA249
    Dim objEllipseSegment As HMIEllipseSegment
    Set objEllipseSegment = ActiveDocument.HMIObjects("EllipseSegment")
    objEllipseSegment.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA250
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ToolTipText Property (Page 2445)
BackFlashColorOn Property (Page 2139)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
StartAngle Property (Page 2426)
RadiusWidth Property (Page 2395)
RadiusHeight Property (Page 2394)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
EndAngle Property (Page 2222)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
Event Object

Description

Represents an event that triggers one or more actions in Runtime (e.g. a direct connection). An event can be configured onto an object and a property.

VBA Object Name

HMIEvent

Usage

Use the AddAction method to configure an action on an event. In this example a C action is to be triggered in the event of a change of radius in Runtime:

Sub AddActionToPropertyTypeCScript()
    'VBA251
    Dim objEvent As HMIEvent
    Dim objCScript As HMIScriptInfo
    Dim objCircle As HMICircle
    'Create circle in the picture. If property "Radius" is changed, 'a C-action is added:
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
    Set objEvent = objCircle.Radius.Events(1)
    Set objCScript = objEvent.Actions.AddAction(hmiActionCreationTypeCScript)
End Sub

See also

Application Property (Page 2121)
Delete Method (Page 1831)
AddAction Method (Page 1794)
VBA Reference (Page 1745)
Parent Property (Page 2359)
EventType Property (Page 2225)
Actions Property (Page 2106)
EventName property (Page 2225)

Events Object (Listing)

Description

- You configure an action on a property with VBA by using the "Events(1)" property, where the index "1" stands for the event "Upon change":
- To configure an action onto an object with the aid of VBA, use the "Events(Index)" property, where "Index" stands for the trigger event (see table):

<table>
<thead>
<tr>
<th>Index</th>
<th>EventType (depending upon the object used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hmiEventTypeNotDefined</td>
</tr>
<tr>
<td>1</td>
<td>hmiEventTypeMouseClick</td>
</tr>
<tr>
<td>2</td>
<td>hmiEventTypeMouseButtonDown</td>
</tr>
<tr>
<td>3</td>
<td>hmiEventTypeMouseButtonUp</td>
</tr>
<tr>
<td>4</td>
<td>hmiEventTypeMouseButtonRButtonDown</td>
</tr>
<tr>
<td>5</td>
<td>hmiEventTypeMouseButtonRButtonUp</td>
</tr>
<tr>
<td>6</td>
<td>hmiEventTypeKeyboardDown</td>
</tr>
<tr>
<td>7</td>
<td>hmiEventTypeKeyUp</td>
</tr>
<tr>
<td>8</td>
<td>hmiEventTypeFocusEnter</td>
</tr>
<tr>
<td>9</td>
<td>hmiEventTypeObjectChange</td>
</tr>
<tr>
<td>10</td>
<td>hmiEventTypePictureOpen</td>
</tr>
</tbody>
</table>

VBA Object Name

HMIEvents
Usage

Use the Item method to return an individual Event object. In this example the event names and event types of all objects in the active pictures are put out. In order for this example to work, insert some objects into the active picture and configure different events.

Sub ShowEventsOfAllObjectsInActiveDocument()
'VBA252
Dim colEvents As HMIEvents
Dim objEvent As HMIEvent
Dim iMax As Integer
Dim iIndex As Integer
Dim iAnswer As Integer
Dim strEventName As String
Dim strObjectName As String
Dim varEventType As Variant
iIndex = 1
iMax = ActiveDocument.HMIObjects.Count
For iIndex = 1 To iMax
Set colEvents = ActiveDocument.HMIObjects(iIndex).Events
strObjectName = ActiveDocument.HMIObjects(iIndex).ObjectName
For Each objEvent In colEvents
strEventName = objEvent.EventName
varEventType = objEvent.EventType
iAnswer = MsgBox("Objectname: " & strObjectName & vbCrLf & "Eventtype: " & varEventType & vbCrLf & "Eventname: " & strEventName, vbOKCancel)
If vbCancel = iAnswer Then Exit For
Next objEvent
If vbCancel = iAnswer Then Exit For
Next iIndex
End Sub

See also

Item Method (Page 1859)
VBA Reference (Page 1745)
Parent Property (Page 2359)
Count Property (Page 2201)
Application Property (Page 2121)

ExternalShapeInfo object

Description

Is used for internal purposes in Graphics Designer.
FaceplateObject object

Description

Represents the “faceplate instance” object. The FaceplateObject object is an element of the following lists:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all default, smart, window and tube objects.

VBA object name

HMIFaceplateObject

Usage

Use the Add method to create a new “faceplate instance” object in a picture:

```vba
Sub AddFaceplateInstance()
'VBA826
Dim objFaceplateInstance As HMIFaceplateObject
Set objFaceplateInstance = ActiveDocument.HMIObjects.AddHMIObject("faceplate instance", "HMIFaceplateObject")
objFaceplateInstance.Properties.Item(3).value = "Faceplate1.fpt"
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditFaceplateInstance()
'VBA827
Dim objFaceplateInstance As HMIFaceplateObject
Set objFaceplateInstance = ActiveDocument.HMIObjects("faceplate instance")
objFaceplateInstance.visible = True
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
    'VBA828
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

- ObjectName Property (Page 2347)
- Layer Property (Page 2293)
- Left Property (Page 2305)
- Top Property (Page 2446)
- Width Property (Page 2553)
- Height Property (Page 2269)
- Operation Property (Page 2353)
- PasswordLevel Property (Page 2362)
- Visible Property (Page 2551)
- ToolTipText Property (Page 2445)
- ScalingMode property (Page 2405)
- FaceplateType property (Page 2228)
- Delete Method (Page 1831)
- Destroy Method (Page 1835)
- Application Property (Page 2121)
- Events Property (Page 2223)
- GroupParent Property (Page 2268)
- InheritState property (Page 2278)
- LDTooltipTexts Property (Page 2303)
- Parent Property (Page 2359)
- Properties Property (Page 2388)
- Selected Property (Page 2412)
- TabOrderSwitch Property (Page 2433)
- TabOrderAlpha Property (Page 2429)
- Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)

FaceplateObjects object

Description
A listing of the HMIFaceplateObject objects that represent all faceplate objects in the picture.

VBA Object Name
HMIFaceplateObjects

See also
Parent Property (Page 2359)
OriginalPropertyName property (Page 2357)

FaceplateProperty object

Description
Represents the property of a faceplate object. In the case of the FaceplateProperty object, the use of the Value property is set as the default. For this reason you can use the following notation, for example, to assign a new value to an object property:

<FaceplateObject>.<FaceplateProperty> = <Value>

You can use the "Dynamic" property to make an object property dynamic with VBA. Use the "Events" listing to configure actions with VBA.

See also
CreateDynamic Method (Page 1828)
deleteDynamic Method (Page 1833)
Application Property (Page 2121)
DisplayName Property (Page 2214)
Dynamic Property (Page 2218)
Events Property (Page 2223)
isDynamicable Property (Page 2280)
LDFonts Property (Page 2298)
LDTexts Property (Page 2303)
Name Property (Page 2343)
FolderItem Object

Description

Represented a folder or object in the Components Library. A FolderItem object of the "Folder" type is an element of the FolderItems listing. A FolderItem object of the "Item" type is an element of the Folder listing.

VBA Object Name

HMIFolderItem

Usage

Use the FolderItems property to return the FolderItems listing. In the following example the names of folders in the "Global Library" will be output:

```
Sub ShowFolderItemsOfGlobalLibrary()
    'VBA253
    Dim colFolderItems As HMIFolderItems
    Dim objFolderItem As HMIFolderItem
    Set colFolderItems = Application.SymbolLibraries(1).FolderItems
    For Each objFolderItem In colFolderItems
        MsgBox objFolderItem.Name
    Next objFolderItem
End Sub
```
Use the CopyToClipboard method to copy a "FolderItem" object of the "Item" type to the clipboard. In the following example the object "PC" will be copied to the clipboard.

```vba
Sub CopyFolderItemToClipboard()
    'VBA254
    Dim objGlobalLib As HMISymbolLibrary
    Set objGlobalLib = Application.SymbolLibraries(1)
    objGlobalLib.FolderItems("Folder2").Folder("Folder2").Folder.Item("Object1").CopyToClipboard
End Sub
```

See also

- Type Property (Page 2450)
- FolderItems Object (Listing) (Page 1973)
- Delete Method (Page 1831)
- CopyToClipboard Method (Page 1824)
- How to paste an object from the object library into a picture with VBA (Page 1652)
- How to edit the component library with VBA (Page 1649)
- VBA Reference (Page 1745)
- Accessing the component library with VBA (Page 1646)
- Parent Property (Page 2359)
- Name Property (Page 2343)
- LDNames Property (Page 2301)
- Folder Property (Page 2252)
- Application Property (Page 2121)
- DisplayName Property (Page 2214)
- Pathname Property (Page 2364)

FolderItems Object (Listing)

Description
A listing of the FolderItem objects that represent all the folders and objects in the Components Library.

**VBA Object Name**

HMIFolderItems

**Usage**

Use the FolderItems property to return the FolderItems listing. In the following example the names of folders in the "Global Library will be output:

```vba
Sub ShowFolderItemsOfGlobalLibrary()
    'VBA255
    Dim colFolderItems As HMIFolderItems
    Dim objFolderItem As HMIFolderItem
    Set colFolderItems = Application.SymbolLibraries(1).FolderItems
    For Each objFolderItem In colFolderItems
        MsgBox objFolderItem.Name
    Next objFolderItem
End Sub
```

Use the AddFolder method, for instance, to create a new folder in the Components Library. In the following example the folder "Project Folder" will be created in the "Project Library":

```vba
Sub AddNewFolderToProjectLibrary()
    'VBA256
    Dim objProjectLib As HMISymbolLibrary
    Set objProjectLib = Application.SymbolLibraries(2)
    objProjectLib.FolderItems.AddFolder("My Folder")
End Sub
```

**See also**

AddItem Method (Page 1802)
SymbolLibrary Object (Page 2074)
FolderItem Object (Page 1972)
AddFromClipboard Method (Page 1799)
AddFolder Method (Page 1798)
How to paste an object from the object library into a picture with VBA (Page 1652)
How to edit the component library with VBA (Page 1649)
VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)
Parent Property (Page 2359)
GraphicObject Object

Description

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIGraphicObject

Usage

Use the Add method to create a new "Graphic Object" object in a picture:

```vba
Sub AddGraphicObject()
    'VBA257
    Dim objGraphicObject As HMIGraphicObject
    Set objGraphicObject = ActiveDocument.HMIOjects.AddHMIObject("Graphic-Object", "HMIGraphicObject")
End Sub
```
Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditGraphicObject()
    'VBA258
    Dim objGraphicObject As HMIGraphicObject
    Set objGraphicObject = ActiveDocument.HMIObjects("Graphic-Object")
    objGraphicObject.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA259
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name of the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

Left Property (Page 2305)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
PicUseTransColor Property (Page 2377)
PictureName Property (Page 2373)
PicTransColor Property (Page 2371)
PicReferenced Property (Page 2370)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Name Property (Page 2343)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
Group Object

Description

Represents the object called "Group Object". The Group Object is an element of the following listings:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.

A group object is created from the objects selected in a picture. The objects in the Group Object are also saved in the "GroupedHMIObjects" listing and index numbers are newly allocated.

You have unrestricted access to the properties of all objects in the Group Object.

Further information regarding group objects can be found in the WinCC documentation under "Group Object".

VBA Object Name

**HMIGroup**

Usage

Use the CreateGroup Method with the Selection listing to create a new "Group Object" object in a picture:

```vba
Sub DoCreateGroup()
'VBA260
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.Selection.CreateGroup
objGroup.ObjectName = "Group-Object"
End Sub
```

Use the following methods to edit an existing Group Object:

- **Methode "Add(Index)"**: Adds a new object to the group object.
- **Methode "Remove(Index)"**: Removes a object from the group object.
- "UnGroup()" method: Ungroups the group object (ungroup).
- "Delete()" Method: Deletes the group object and the objects that it contains.

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditGroup()
    'VBA261
    Dim objGroup As HMIGroup
    Set objGroup = ActiveDocument.HMIObjects("Group-Object")
    MsgBox objGroup.ObjectName
End Sub
```

See also

- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- GroupedObjects Object (Listing) (Page 1983)
- Ungroup Method (Page 1892)
- Remove Method (Page 1871)
- Delete Method (Page 1831)
- Add Method (GroupedObjects Listing) (Page 1791)
- How to Edit Objects in Group Objects Using VBA (Page 1679)
- How to Edit the Group Objects Using VBA (Page 1676)
- VBA Reference (Page 1745)
- Group Objects (Page 1674)
- Editing Objects with VBA (Page 1660)
- Application Property (Page 2121)
- Events Property (Page 2223)
- GroupParent Property (Page 2268)
- GroupedHMIObjects Property (Page 2269)
- Height Property (Page 2269)
- Layer Property (Page 2293)
- LDTooltipTexts Property (Page 2303)
- Left Property (Page 2305)
- ObjectName Property (Page 2347)
- Operation Property (Page 2353)
- Parent Property (Page 2359)
GroupDisplay Object

Description

Represents the "Group Display" object. The GroupDisplay object is an element of the following listings:
- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

VBA Object Name

HMIGroupDisplay
Usage

Use the Add method to create a new "Group Display" object in a picture:

Sub AddGroupDisplay()
    'VBA262
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("Groupdisplay", "HMIGroupDisplay")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditGroupDisplay()
    'VBA263
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects("Groupdisplay")
    objGroupDisplay.BackColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA264
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

MCText Property (Page 2336)
Height Property (Page 2269)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
UserValue1 Property (Page 2466)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
SignificantMask Property (Page 2419)
SameSize Property (Page 2403)
Relevant Property (Page 2398)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
MessageClass Property (Page 2339)
MCGUTextFlash Property (Page 2327)
MCGUTextColorOn Property (Page 2326)
MCGUTextColorOff Property (Page 2326)
MCGUBackFlash Property (Page 2325)
MCGUBackColorOn Property (Page 2324)
MCGUBackColorOff-Eigenschaft (Page 2324)
LockText Property (Page 2316)
LockTextColor Property (Page 2317)
LockStatus Property (Page 2315)
LockBackColor Property (Page 2313)
Left Property (Page 2305)
Layer Property (Page 2293)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FontBold Property (Page 2256)
FlashRate Property (Page 2246)
Button1..8Width property (Page 2166)
BackColor Property (Page 2131)
BackBorderWidth Property (Page 2131)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
GroupedObjects Object (Listing)

Description

A listing of the HMIObject objects that represent all the objects in the group object.

VBA Object Name

HMIGroupedObjects
Usage

Use the GroupedHMIObjects property to return the GroupedObjects listing. In the following example all the objects in the first group object are output in the active picture. The group object called "Group1" must first have been created:

Sub ShowGroupedObjectsOfFirstGroup()
  'VBA265
  Dim colGroupedObjects As HMIGroupedObjects
  Dim objObject As HMIObject
  Set colGroupedObjects = ActiveDocument.HMIObjects("Group1").GroupedHMIObjects
  For Each objObject In colGroupedObjects
    MsgBox objObject.ObjectName
  Next objObject
End Sub

Use the Remove method, for instance, to remove an object from the group object. In the following example the first object will be removed from the group object called "Group1":

Sub RemoveObjectFromGroup()
  'VBA266
  Dim objGroup As HMIGroup
  Set objGroup = ActiveDocument.HMIObjects("Group1")
  objGroup.GroupedHMIObjects.Remove (1)
End Sub

See also

Group Object (Page 1978)
Remove Method (Page 1871)
Add Method (GroupedObjects Listing) (Page 1791)
How to Edit the Group Objects Using VBA (Page 1676)
VBA Reference (Page 1745)
Group Objects (Page 1674)
Parent Property (Page 2359)
GroupedHMIObjects Property (Page 2269)
Count Property (Page 2201)
Application Property (Page 2121)
Item Property (Page 2283)
### HMIDefaultObjects Object (Listing)

#### Description

![Diagram of HMIDefaultObjects](image)

A listing of the following HMIObject objects:

<table>
<thead>
<tr>
<th>Object</th>
<th>VBA object name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>HMILine</td>
</tr>
<tr>
<td>Polygon</td>
<td>HMIPolygon</td>
</tr>
<tr>
<td>Polyline</td>
<td>HMIPolyLine</td>
</tr>
<tr>
<td>Ellipse</td>
<td>HMIEllipse</td>
</tr>
<tr>
<td>Circle</td>
<td>HMICircle</td>
</tr>
<tr>
<td>Ellipse segment</td>
<td>HMIEllipseSegment</td>
</tr>
<tr>
<td>Pie segment</td>
<td>HMIPieSegment</td>
</tr>
<tr>
<td>Ellipse arc</td>
<td>HMIEllipseArc</td>
</tr>
<tr>
<td>Circular arc</td>
<td>HMICompoundArc</td>
</tr>
<tr>
<td>Rectangle</td>
<td>HMIRectangle</td>
</tr>
<tr>
<td>Rounded rectangle</td>
<td>HMIBooleanRectangle</td>
</tr>
<tr>
<td>Application window</td>
<td>HMIApplicationWindow</td>
</tr>
<tr>
<td>Screen Window</td>
<td>HMIStaticText</td>
</tr>
<tr>
<td>I/O Field</td>
<td>HMIIOField</td>
</tr>
<tr>
<td>Button</td>
<td>HMIButton</td>
</tr>
<tr>
<td>Check box</td>
<td>HMICheckBox</td>
</tr>
<tr>
<td>Radio box</td>
<td>HMIOptionGroup</td>
</tr>
<tr>
<td>Round button</td>
<td>HMIRoundButton</td>
</tr>
<tr>
<td>Bar</td>
<td>HMIBooleanBarGraph</td>
</tr>
<tr>
<td>Slider object</td>
<td>HMIIntegerSlider</td>
</tr>
<tr>
<td>Graphic Object</td>
<td>HMIStyle</td>
</tr>
<tr>
<td>Status display</td>
<td>HMIStatusDisplay</td>
</tr>
<tr>
<td>Text list</td>
<td>HMITextList</td>
</tr>
<tr>
<td>Connector</td>
<td>HMIObjectConnection</td>
</tr>
<tr>
<td>Multiple row text</td>
<td>HMIMultiLineEdit</td>
</tr>
<tr>
<td>Combo box</td>
<td>HMIComboBox</td>
</tr>
<tr>
<td>List box</td>
<td>HMILListBox</td>
</tr>
<tr>
<td>Polygon tube</td>
<td>HMITubePolyline</td>
</tr>
<tr>
<td>T-piece</td>
<td>HMITubeTeeObject</td>
</tr>
</tbody>
</table>
**VBA for Automated Configuration**

### 3.6 VBA Reference

<table>
<thead>
<tr>
<th>Object</th>
<th>VBA object name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double T-piece</td>
<td>HMITubeDoubleTeeObject</td>
</tr>
<tr>
<td>Tube bend</td>
<td>HMITubeArcObject</td>
</tr>
<tr>
<td>3D bar</td>
<td>HMI3DBarGraph</td>
</tr>
<tr>
<td>Group display</td>
<td>HMIGroupDisplay</td>
</tr>
<tr>
<td>Faceplate instance</td>
<td>HMIFaceplateObject</td>
</tr>
</tbody>
</table>

**VBA object name**

HMIDefaultObjects

**Usage**

Use the DefaultHMIObjects property to change the default property values of the included objects. In this example all the objects contained in the listing will be output:

```vba
Sub ShowDefaultObjects()
'VBA267
Dim strType As String
Dim strName As String
Dim strMessage As String
Dim iMax As Integer
Dim iIndex As Integer
iMax = Application.DefaultHMIObjects.Count
iIndex = 1
For iIndex = 1 To iMax
    With Application.DefaultHMIObjects(iIndex)
        strType = .Type
        strName = .ObjectName
        strMessage = strMessage & "Element: " & iIndex & " / Objecttype: " & strType & " / Objectname: " & strName
    End With
    If 0 = iIndex Mod 10 Then
        MsgBox strMessage
        strMessage = ""
    Else
        strMessage = strMessage & vbCrLf & vbCrLf
    End If
Next iIndex
MsgBox "Element: " & iIndex & vbCrLf & "Objecttype: " & strType & vbCrLf & "Objectname: " & strName
End Sub
```

**See also**

Button Object (Page 1921)

TextList Object (Page 2076)

StatusDisplay Object (Page 2070)
StaticText Object (Page 2067)
Slider object (Page 2062)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
PolyLine Object (Page 2038)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
PictureWindow Object (Page 2029)
OptionGroup Object (Page 2025)
Line Object (Page 2004)
IOField Object (Page 1992)
EllipseSegment Object (Page 1963)
EllipseArc Object (Page 1960)
Ellipse Object (Page 1957)
CircularArc Object (Page 1931)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
BarGraph Object (Page 1914)
ApplicationWindow Object (Page 1911)
VBA Reference (Page 1745)
Parent Property (Page 2359)
Count Property (Page 2201)
DefaultHMIObjects Property (Page 2210)
Application Property (Page 2121)
Item Property (Page 2283)
HMIObject Object

Description

Represents an object from the Object Palette of the Graphics Designer. The HMIObject object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

This object contains the object properties that apply to all standard, smart and Windows objects (incl. Width, Height, Top and Left).

VBA Object Name

HMIObject

Usage

Use HMIObjects(Index), for instance, to return an individual HMIObject object. "For Index you can use either the index number or the name of the object. In the following example the name of the first object in the active picture is output:

Sub ShowFirstObjectOfCollection()
    'VBA268
    Dim strName As String
    strName = ActiveDocument.HMIObjects(1).ObjectName
    MsgBox strName
End Sub
Use the Delete method to remove an object from the HMIObjects listing. In the following example the first object in the active picture will be removed:

Sub DeleteObject()
  'VBA269
  ActiveDocument.HMIObjects(1).Delete
End Sub

See also

- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- HMI_DefaultObjects Object (Listing) (Page 1985)
- Delete Method (Page 1831)
- VBA Reference (Page 1745)
- Default objects, Smart objects, Windows objects and Tube objects (Page 1662)
- Editing Objects with VBA (Page 1660)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Type Property (Page 2450)
- Top Property (Page 2446)
- ToolTipText Property (Page 2445)
- TabOrderAlpha Property (Page 2429)
- TabOrderSwitch Property (Page 2433)
- Selected Property (Page 2412)
- Properties Property (Page 2388)
- PasswordLevel Property (Page 2362)
- Parent Property (Page 2359)
- Operation Property (Page 2353)
- Left Property (Page 2305)
- LDTooltipTexts Property (Page 2303)
- Layer Property (Page 2293)
- Height Property (Page 2269)
- GroupParent Property (Page 2268)
- Events Property (Page 2223)
- Application Property (Page 2121)
ObjectName Property (Page 2347)
ConnectionPoints property (Page 2198)

**HMIObjects Object (Listing)**

**Description**

A listing of the HMIObject objects that represent all the objects in the picture.

**VBA Object Name**

HMIObjects

**Note**

The sequence of HMI objects in the HMIObjects list can be altered by adding and/or deleting HMI objects.

The sequence of listing can also change if HMI objects are processed in the current listing. This behavior can occur if the Layers property is modified and/or if the methods "SendToBack" and "BringToFront" are used.

**Usage**

Use the HMIObjects property to return the HMIObjects listing. In the following example all the object names in the active picture are output:

```vba
Sub ShowObjectsOfDocument()
  'VBA270
  Dim colObjects As HMIObjects
  Dim objObject As HMIObject
  Set colObjects = ActiveDocument.HMIObjects
  For Each objObject In colObjects
    MsgBox objObject.ObjectName
  Next objObject
End Sub
```
Use the `AddHMIObject` method to create a new object in the picture. In the following example, a circle will be inserted into the active picture:

```vba
Sub AddCircle()
  'VBA271
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_1", "HMICircle")
End Sub
```

Use the `Find` method to search for one or more objects in the picture. In the following example, objects of the "HMICircle" type will be searched for in the active picture:

```vba
Sub FindObjectsByType()
  'VBA272
  Dim colSearchResults As HMICollection
  Dim objMember As HMIObject
  Dim iResult As Integer
  Dim strName As String
  Set colSearchResults = ActiveDocument.HMIObjects.Find(ObjectType:="HMICircle")
  For Each objMember In colSearchResults
    iResult = colSearchResults.Count
    strName = objMember.ObjectName
    MsgBox "Found: " & CStr(iResult) & vbCrLf & "Objectname: " & strName
  Next objMember
End Sub
```

See also

- Count Property (Page 2201)
- HMIDefaultObjects Object (Listing) (Page 1985)
- SelectedObjects object (Listing) (Page 2060)
- Find Method (Page 1841)
- AddOLEObject Method (Page 1803)
- AddHMIObject Method (Page 1801)
- AddActiveXControl Method (Page 1795)
- How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1664)
- VBA Reference (Page 1745)
- Default objects, Smart objects, Windows objects and Tube objects (Page 1662)
- Editing Objects with VBA (Page 1660)
- Parent Property (Page 2359)
- Application Property (Page 2121)
- AddDotNetControl method (Page 1797)
IOField Object

Description

Represents the "I/O Field" object. The IOField object is an element of the following listings:

- **Objects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIIOField

Usage

Use the Add method to create a new "I/O Field" object in a picture:

```vba
Sub AddIOField()
    'VBA273
    Dim objIOField As HMIIOField
    Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IO-Field", "HMIIOField")
End Sub
```
Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditIOField()
'VBA274
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects("IO-Field")
objIOField.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA275
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

Use the "HMIDefaultObjects(Index)" to return an object from the HMIDefaultObjects Listing:

Sub EditDefaultPropertiesOfIOField()
'VBA276
Dim objIOField As HMIIOField
Set objIOField = Application.DefaultHMIObjects("HMIIOField")
objIOField.BorderColor = RGB(255, 255, 0)
End Sub

See also

LimitMin Property (Page 2311)
ClearOnNew Property (Page 2181)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
PasswordLevel Property (Page 2362)
OutputValue Property (Page 2358)
OutputFormat Property (Page 2357)
Orientation Property (Page 2356)
OperationReport Property (Page 2355)
OperationMessage Property (Page 2354)
Operation Property (Page 2353)
LimitMax Property (Page 2310)
Left Property (Page 2305)
Layer Property (Page 2293)
HiddenInput Property (Page 2271)
Height Property (Page 2269)
ForeFlashColorOn Property (Page 2262)
ForeFlashColorOff Property (Page 2261)
ForeColor Property (Page 2260)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FontBold Property (Page 2256)
FlashRateForeColor Property (Page 2251)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashForeColor Property (Page 2242)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
EditAtOnce Property (Page 2219)
DataFormat Property (Page 2210)
CursorControl Property (Page 2206)
ClearOnError Property (Page 2180)
BoxType Property (Page 2165)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
AssumeOnFull Property (Page 2123)
AssumeOnExit Property (Page 2123)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
AdaptBorder Property (Page 2111)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
InputValue property (Page 2279)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
LanguageFont Object

Description

Contains the font settings for the project language. The LanguageFont object is an element of the LanguageFonts listing.

VBA Object Name

HMILanguageFont

Usage

Use LDFonts(Index) to return an individual LanguageFont object. In the following example a Button object will be created and the name of the first configured font will be output:

```vba
Sub ShowFirstObjectOfCollection()
    'VBA277
    Dim strName As String
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button", "HMIButton")
    strName = objButton.LDFonts(1).Family
    MsgBox strName
End Sub
```

Object properties

The LanguageFont object possesses the following properties:

See also

LanguageFonts Object (Listing) (Page 1997)
VBA Reference (Page 1745)
Underlined Property (Page 2460)
Size Property (Page 2419)
Parent Property (Page 2359)
LanguageID Property (Page 2289)
Italic Property (Page 2282)
Family Property (Page 2229)
Bold Property (Page 2151)
Application Property (Page 2121)

LanguageFonts Object (Listing)

Description

A listing of the LanguageFont objects that represent all the language-dependent fonts in an object.

VBA Object Name

HMILanguageFonts

Usage

Use the LDFonts property to return the LanguageFonts listing. In the following example the language identifiers of the configured fonts will be output:

Sub ShowLanguageFont()
    'VBA278
    Dim colLanguageFonts As HMILanguageFonts
    Dim objLanguageFont As HMILanguageFont
    Dim objButton As HMIButton
    Dim iMax As Integer
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    Set colLanguageFonts = objButton.LDFonts
    iMax = colLanguageFonts.Count
    For Each objLanguageFont In colLanguageFonts
        MsgBox "Planned fonts: " & iMax & vbCrLf & "Language-ID: " & objLanguageFont.LanguageID
    Next objLanguageFont
End Sub
Use the ItemByLcid method to define the language for which it is intended to enter font settings. The following example sets the font attributes of a button for French and English.

**Note**

For this example to work, you must already have configured in the languages concerned.

```vba
Sub ExampleForLanguageFonts()
    'VBA279
    Dim colLangFonts As HMILanguageFonts
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    objButton.Text = "DefText"
    Set colLangFonts = objButton.LDFonts

    'Adjust fontsettings for french:
    With colLangFonts.ItemByLCID(1036)
        .Family = "Courier New"
        .Bold = True
        .Italic = False
        .Underlined = True
        .Size = 12
    End With

    'Adjust fontsettings for english:
    With colLangFonts.ItemByLCID(1033)
        .Family = "Times New Roman"
        .Bold = False
        .Italic = True
        .Underlined = False
        .Size = 14
    End With
End Sub
```

See also

- LanguageFont Object (Page 1996)
- ItemByLcid Method (Page 1861)
- Item Method (Page 1859)
- VBA Reference (Page 1745)
- Parent Property (Page 2359)
- Count Property (Page 2201)
- Application Property (Page 2121)
LanguageText Object

Description

Contains the multilingual labels for an object. The LanguageText object is an element of the LanguageTexts listing.

VBA Object Name

HMI.LanguageText

Usage

In the following example a German label and an English label will be assigned to the button called "myButton":

```vba
Sub AddLanguagesToButton()
'VBA280
Dim objLabelText As HMI.LanguageText
Dim objButton As HMI.Button
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMI.Button")
',
'Add text in actual datalanguage:
objButton.Text = "Actual-Language Text"
',
'Add english text:
Set objLabelText = ActiveDocument.HMIObjects("myButton").LDTexts.Add(1033, "English Text")
End Sub
```

See also

LanguageTexts Object (Listing) (Page 2000)
Delete Method (Page 1831)
VBA Reference (Page 1745)
Parent Property (Page 2359)
LanguageID Property (Page 2289)
LanguageTexts Object (Listing)

Description

A listing of the LanguageText objects that represent all the multilingual texts in an object.

VBA Object Name

HMILanguageTexts

Usage

Use one of the following properties to return the LanguageTexts listing:

- LDLabelTexts Property
- LDNames Property
- LDStatusTexts Property
- LDTeks Property
- LDLToolTipTexts Property

An example showing how to use the LanguageTexts listing can be found in this documentation under the heading “LDStatusTexts Property”.
Use the Add method to add multilingual texts to an object. In the following example a German label and an English label will be assigned to the button called "myButton":

Sub AddLanguagesToButton()
' VBA281
Dim objLabelText As HMILanguageText
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
',
' Add text in actual datalanguage:
objButton.Text = "Actual-Language Text"
',
' Add english text:
Set objLabelText = ActiveDocument.HMIObjects("myButton").LDTexts.Add(1033, "English Text")
End Sub

See also
LanguageText Object (Page 1999)
ItemByLcid Method (Page 1861)
Item Method (Page 1859)
VBA Reference (Page 1745)
Parent Property (Page 2359)
LDTooltipTexts Property (Page 2303)
LDTexts Property (Page 2303)
LDStatusTexts Property (Page 2301)
LDNames Property (Page 2301)
LDLabelTexts Property (Page 2299)
Count Property (Page 2201)
Application Property (Page 2121)

Layer Object

Description
Represents one of the 32 layers that are available in the picture.

**VBA Object Name**

HMILayer

**Usage**

Use the Layer object to define a name and the minimum and maximum zoom for a layer. You define the visibility of layers separately by CS and RT layers:

- Document Object: Controls the visibility of the RT layers.
- View Object: Controls the visibility of the RT layers.

Use the Layers listing to return a Layer object. In the following example the settings for the lowest layer are configured in the active picture:

```vba
Sub ConfigureSettingsOfLayer()
    'VBA282
    Dim objLayer As HMILayer
    Set objLayer = ActiveDocument.Layers(1)
    With objLayer
        'configure "Layer 0"
        .MinZoom = 10
        .MaxZoom = 100
        .Name = "Configured with VBA"
    End With
End Sub
```

**See also**

- Layers Property (Page 2297)
- VBA Reference (Page 1745)
- Editing Layers with VBA (Page 1657)
- Visible Property (Page 2551)
- Number Property (Page 2345)
- Name Property (Page 2343)
- MinZoom Property (Page 2340)
- MaxZoom Property (Page 2323)
- LDNames Property (Page 2301)
- ActiveLayer Property (Page 2108)
Layers Object (Listing)

Description

A listing of the Layer objects that represent the 32 layers in the picture.

VBA Object Name

HMILayer

Usage

Use the LayersCS or LayersRT property to return the Layers listing. In the following example the layer names in the copy of the active picture will be output:

```
Sub ShowLayer()
    'VBA283
    Dim colLayers As HMILayers
    Dim objLayer As HMILayer
    Dim strLayerList As String
    Dim iCounter As Integer
    iCounter = 1
    Set colLayers = ActiveDocument.Layers
    For Each objLayer In colLayers
        If 1 = iCounter Mod 2 And 32 > iCounter Then
            strLayerList = strLayerList & vbCrLf
        ElseIf 11 > iCounter Then
            strLayerList = strLayerList & "       
        Else
            strLayerList = strLayerList & "     
        End If
        strLayerList = strLayerList & objLayer.Name
        iCounter = iCounter + 1
    Next objLayer
    MsgBox strLayerList
End Sub
```

See also

Layer Object (Page 2001)
Item Method (Page 1859)
Line Object

Description

Represents the "Line" object. The Line object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMI_DEFAULT_OBJECTS: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMILine

Usage

Use the Add method to create a new "Line" object in a picture:

Sub AddLine()
  'VBA285
  Dim objLine As HMILine
  Set objLine = ActiveDocument.HMIObjects.AddHMIObject("Line1", "HMILine")
End Sub
Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditLine()
    'VBA286
    Dim objLine As HMILine
    Set objLine = ActiveDocument.HMIObjects("Lin1")
    objLine.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA287
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name of the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

AddHMIObject Method (Page 1801)
BorderColorBackColor Property (Page 2152)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIObjects Object (Listing) (Page 1990)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
RotationAngle Property (Page 2400)
ReferenceRotationTop Property (Page 2397)
ReferenceRotationLeft Property (Page 2396)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Index Property (Page 2276)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashBorderColor Property (Page 2240)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderEndStyle Property (Page 2156)
BorderColor Property (Page 2153)
ActualPointTop Property (Page 2110)
ActualPointLeft Property (Page 2109)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
ListBox object

Description

Represents the "ListBox" object. The ListBox object is an element of the following listings:
- HMIObjec: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all default, smart, window and tube objects.

VBA object name

HMIListBox

Usage

Use the Add method to create a new "ListBox" object in a picture:

Sub AddListBox()
'VBA829
Dim objListBox As HMIListBox
Set objListBox = ActiveDocument.HMIObjec.AddHMIObject("ListBox", "HMIListBox")
End Sub

Use "HMIObjec"(Index)" to return an object from the HMIObjec listing, where Index in this case identifies the object by name:

Sub EditListBox()
'VBA830
Dim objListBox As HMIListBox
Set objListBox = ActiveDocument.HMIObjec("ListBox")
objListBox.BorderColor = RGB(255, 0, 0)
End Sub
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA831
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ObjectName Property (Page 2347)
Layer Property (Page 2293)
Left Property (Page 2305)
Top Property (Page 2446)
Width Property (Page 2553)
Height Property (Page 2269)
NumberLines Property (Page 2346)
ForeColor Property (Page 2260)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackColor Property (Page 2131)
FillColor Property (Page 2231)
BorderStyle Property (Page 2160)
BorderWidth Property (Page 2161)
FillStyle Property (Page 2235)
GlobalShadow property (Page 2264)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontUnderline Property (Page 2259)
AlignmentLeft Property (Page 2117)
Index Property (Page 2276)
Text Property (Page 2439)
Operation Property (Page 2353)
Menu Object

Description

Represents the "User Defined Menu" object. The Menu object is an element of the CustomMenus listing.
VBA Object Name

HMIMenu

Usage

Use CustomMenus(Index) to return an individual Menu object. "For Index you can use either
the index number or the name of the object. In order for the following example to work, create
a user defined menu. For an example of this, please refer to "Creating a New Application-
Specific Menu" in this documentation. In the following example the name of the first user-
defined menu in the active picture will be output:

Sub ShowFirstMenuOfMenucollection()
  'VBA288
  Dim strName As String
  strName = ActiveDocument.CustomMenus(1).Label
  MsgBox strName
End Sub

Use the Delete method to remove a "Menu" object from the "CustomMenus" listing. In the
following example the first user-defined menu in the active picture will be removed:

Sub DeleteMenu()
  'VBA289
  Dim objMenu As HMIMenu
  Set objMenu = ActiveDocument.CustomMenus(1)
  objMenu.Delete
End Sub

See also

Menus Object (Listing) (Page 2011)
Delete Method (Page 1831)
How to Create Picture-specific Menus and Toolbars (Page 1655)
How to Create a New Application-Specific Menu (Page 1629)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)
Visible Property (Page 2551)
StatusText Property (Page 2426)
Position Property (Page 2380)
Parent Property (Page 2359)
MenuItems Property (Page 2337)
LDStatusTexts Property (Page 2301)
Menus Object (Listing)

Description

A listing of the Menu objects that represent all the user-defined menus in the Graphics Designer.

VBA Object Name

HMIMenus

Usage

Use the CustomMenus property to return the Menus listing. In the following example all the user-defined menus in the active picture will be output.

Note

The Menus listing does not distinguish between application-specific and picture-specific menus in the output.
Sub ShowCustomMenusOfDocument()
'VBA290
Dim colMenus As HMIMenus
Dim objMenu As HMIMenu
Dim strMenuList As String
Set colMenus = ActiveDocument.CustomMenus
For Each objMenu In colMenus
strMenuList = strMenuList & objMenu.Label & vbCrLf
Next objMenu
MsgBox strMenuList
End Sub

Use the Application property and the InsertMenu method if you want to create an application-specific menu. Create the VBA code in either the "Project Template" document or the "Global Template" document. In the following example a user-defined menu called "myApplicationMenu" will be created:

Sub InsertApplicationSpecificMenu()
'VBA291
Dim objMenu As HMIMenu
Set objMenu = Application.CustomMenus.InsertMenu(1, "a_Menu1", "myApplicationMenu")
End Sub

Use the ActiveDocument property and the InsertMenu method if you want to create a picture-specific menu. Create the VBA code in the document called "ThisDocument": In the following example a picture-specific menu called "myDocumentMenu" will be created:

Sub InsertDocumentSpecificMenu()
'VBA292
Dim objMenu As HMIMenu
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "d_Menu1", "myDocumentMenu")
End Sub

See also

Menu Object (Page 2009)
Item Method (Page 1859)
InsertMenu Method (Page 1850)
How to Create Picture-specific Menus and Toolbars (Page 1655)
How to Create a New Application-Specific Menu (Page 1629)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)
Parent Property (Page 2359)
MenuItem Object

Description

Represents a menu entry for a user-defined menu in the Graphics Designer. The MenuItem object is an element of the MenuItems listing.

VBA Object Name

HMIMenuItem

Usage

Note

In order for the examples to work, first create a user-defined menu. For an example of this, please refer to "Adding a New Entry to the Menu" in this documentation.

Use MenuItems(Index) to return an individual MenuItem object. "For Index you can use either the index number or the name of the object. In the following example the first entry in the first user-defined menu in the active picture will be output:

```vba
Sub ShowFirstObjectOfCollection()
  'VBA293
  Dim strName As String
  strName = ActiveDocument.CustomMenus(1).MenuItems(1).Label
  MsgBox strName
End Sub
```
Use the Delete method to remove an object from the "MenuItems" listing. In the following example the first entry in the first user-defined menu in the active picture will be deleted:

```vba
Sub DeleteMenuItem()
    'VBA294
    ActiveDocument.CustomMenus(1).MenuItems(1).Delete
End Sub
```

See also

- Parent Property (Page 2359)
- MenuItems Object (Listing) (Page 2015)
- Delete Method (Page 1831)
- Configuring Menus and Toolbars (Page 1626)
- How to assign VBA macros to menus and toolbars (Page 1642)
- How to assign help texts to menus and toolbars (Page 1639)
- How to add a new menu entry to a menu (Page 1631)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
- Visible Property (Page 2551)
- Tag Property (Page 2434)
- SubMenu Property (Page 2427)
- StatusText Property (Page 2426)
- ShortCut Property (Page 2416)
- Position Property (Page 2380)
- MenuItemType Property (Page 2338)
- Macro Property (Page 2320)
- LDStatusTexts Property (Page 2301)
- LDLabelTexts Property (Page 2299)
- Label Property (Page 2288)
- Key Property (Page 2287)
- Icon Property (Page 2275)
- Enabled Property (Page 2220)
- Checked Property (Page 2172)
- Application Property (Page 2121)
MenuItems Object (Listing)

**Description**

A listing of the MenuItem objects that represent all the entries in a user-defined menu.

**Usage**

**Note**

In order for the examples to work, first create a user-defined menu. For an example of this, please refer to "Adding a New Entry to the Menu" in this documentation.

Use the MenuItems property to return the MenuItems listing. In the following example all the entries in the first user-defined menu in the active picture will be output:

**Note**

The MenuItems listing does not distinguish between an application-specific and a picture-specific menu in the output.

```vba
Sub ShowMenuItems()
    'VBA295
    Dim colMenuItems As HMIMenuItems
    Dim objMenuItem As HMIMenuItem
    Dim strItemList As String
    Set colMenuItems = ActiveDocument.CustomMenus(1).MenuItems
    For Each objMenuItem In colMenuItems
        strItemList = strItemList & objMenuItem.Label & vbCrLf
    Next objMenuItem
    MsgBox strItemList
End Sub
```
Use the InsertMenuItem method, for instance, to insert an entry into an existing user-defined menu. In the following example the picture-specific menu "DocMenu2" will be created in the active picture and the menu entry "MenuItem1" is inserted:

```vba
Sub InsertMenuItem()
    'VBA296
    Dim objMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem
    Set objMenu = ActiveDocument.CustomMenus.InsertMenu(2, "d_Menu2", "DocMenu2")
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "m_Item2_1", "MenuItem 1")
End Sub
```

See also

- InsertSubmenu Method (Page 1854)
- MenuItem Object (Page 2013)
- InsertSeparator Method (Page 1853)
- InsertMenuItem Method (Page 1852)
- How to add a new menu entry to a menu (Page 1631)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
- Parent Property (Page 2359)
- Count Property (Page 2201)
- Application Property (Page 2121)

**MultiLineEdit object**

**Description**
Represents the "MultiLineEdit" object. The MultiLineEdit object is an element of the following listings:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all default, smart, window and tube objects.

**VBA object name**

HMIMultiLineEdit

**Usage**

Use the Add method to create a new "MultiLineEdit" object in a picture:

```
Sub AddMultiLineEdit()
  'VBA832
  Dim objMultiLineEdit As HMIMultiLineEdit
  Set objMultiLineEdit = ActiveDocument.HMIObjects.AddHMIObject("MultiLineEdit",
  "HMIMultiLineEdit")
  End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```
Sub EditMultiLineEdit()
  'VBA833
  Dim objMultiLineEdit As HMIMultiLineEdit
  Set objMultiLineEdit = ActiveDocument.HMIObjects("MultiLineEdit")
  objMultiLineEdit.BorderColor = RGB(255, 0, 0)
  End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

```
Sub ShowNameOfFirstSelectedObject()
  'VBA834
  'Select all objects in the picture:
  ActiveDocument.Selection.SelectAll
  'Get the name from the first object of the selection:
  MsgBox ActiveDocument.Selection(1).ObjectName
  End Sub
```

**See also**

Layer Property (Page 2293)

Left Property (Page 2305)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BorderStyle Property (Page 2160)
BorderWidth Property (Page 2161)
BackColor Property (Page 2131)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontUnderline Property (Page 2259)
ForeColor Property (Page 2260)
AlignmentLeft Property (Page 2117)
Top Property (Page 2446)
Width Property (Page 2553)
Height Property (Page 2269)
Text Property (Page 2439)
Operation Property (Page 2353)
PasswordLevel Property (Page 2362)
Visible Property (Page 2551)
ToolTipText Property (Page 2445)
ObjectName Property (Page 2347)
GlobalShadow property (Page 2264)
Application Property (Page 2121)
GroupParent Property (Page 2268)
LDFonts Property (Page 2298)
LDTexsts Property (Page 2303)
LDTooltipTexts Property (Page 2303)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
**ObjConnection object**

**Description**

 Represents the "Connector" object. The ObjConnection object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

**Note**

You have read-only access to the properties of the ObjConnection object.

**VBA Object Name**

HMIObjConnection

**Application**

From the properties of the ObjConnection object you can find out which objects are connected.

**Example**

In order for the following example to work you must have connected two objects to the connector in the active picture of the Graphics Designer. You can find the Connector object in the Graphics Designer in the Object Palette under "Standard Objects". For this example to work, give the connector the name "Connector1".
In the user-defined menu "Connector Info" you can click on the "Connector Info" entry and display the objects connected via the connector:
Sub ShowConnectorInfo_Menu()
    'VBA297
    Dim objMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem
    Dim strDocName As String
    strDocName = Application.ApplicationDataPath & ActiveDocument.Name
    Set objMenu = Documents(strDocName).CustomMenus.InsertMenu(1, "ConnectorMenu", "Connector_Info")
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "ShowConnectInfo", "Info Connector")
End Sub

Sub ShowConnectorInfo()
    Dim objConnector As HMIObjConnection
    Dim iStart As Integer
    Dim iEnd As Integer
    Dim strStart As String
    Dim strEnd As String
    Dim strObjStart As String
    Dim strObjEnd As String
    Set objConnector = ActiveDocument.HMIObjects("Connector1")
    iStart = objConnector.BottomConnectedConnectionPointIndex
    iEnd = objConnector.TopConnectedConnectionPointIndex
    strObjStart = objConnector.BottomConnectedObjectName
    strObjEnd = objConnector.TopConnectedObjectName
    Select Case iStart
        Case 0
            strStart = "top"
        Case 1
            strStart = "right"
        Case 2
            strStart = "bottom"
        Case 3
            strStart = "left"
    End Select
    Select Case iEnd
        Case 0
            strEnd = "top"
        Case 1
            strEnd = "right"
        Case 2
            strEnd = "bottom"
        Case 3
            strEnd = "left"
    End Select
    MsgBox "The selected connector links the objects " & vbCrLf & strObjStart & vbCrLf & strStart & vbCrLf & strObjEnd & vbCrLf & strEnd & vbCrLf & "Connected points: " & vbCrLf & " & strStart & vbCrLf & strObjStart & ": " & vbCrLf & strStart & vbCrLf & strObjEnd & ": " & vbCrLf & strEnd
End Sub

Private Sub Document_MenuItemClicked(ByVal MenuItem As IHMIMenuItem)
    Select Case MenuItem.Key
        Case "ShowConnectInfo"
            Call ShowConnectorInfo
    End Select
End Sub
See also

- TopConnectedConnectionPointIndex Property (Page 2447)
- TopConnectedObjectName Property (Page 2446)
- BottomConnectedConnectionPointIndex Property (Page 2163)
- BottomConnectedObjectName Property (Page 2162)
- Application Property (Page 2121)
- BorderBackColor Property (Page 2152)
- BorderColor Property (Page 2153)
- BorderEndStyle Property (Page 2156)
- BorderFlashColorOff Property (Page 2157)
- BorderFlashColorOn Property (Page 2159)
- BorderStyle Property (Page 2160)
- BorderWidth Property (Page 2161)
- Events Property (Page 2223)
- FlashBorderColor Property (Page 2240)
- FlashRateBorderColor Property (Page 2249)
- GlobalColorScheme property (Page 2264)
- GlobalShadow property (Page 2264)
- GroupParent Property (Page 2268)
- Height Property (Page 2269)
- Layer Property (Page 2293)
- LDTooltipTexts Property (Page 2303)
- Left Property (Page 2305)
- ObjectName Property (Page 2347)
- Operation Property (Page 2353)
- Orientation Property (Page 2356)
- Parent Property (Page 2359)
- PasswordLevel Property (Page 2362)
- Properties Property (Page 2388)
- Selected Property (Page 2412)
- TabOrderSwitch Property (Page 2433)
- TabOrderAlpha Property (Page 2429)
- ToolTipText Property (Page 2445)
OLEObject Object

Description

![Diagram of OLEObject Object]

Represents the object called "OLE Element". The OLEObject object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.

VBA Object Name

HMIOLEObject

Usage

Use the AddOLEObject method to create a new "OLE Element" object in a picture: In the following example an OLE Element containing a Wordpad document will be inserted into the active picture:

```vba
Sub AddOLEObjectToActiveDocument()
    'VBA298
    Dim objOLEObject As HMIOLEObject
    Set objOLEObject = ActiveDocument.HMIObjects.AddOLEObject("Wordpad Document", "Wordpad.Document.1")
End Sub
```
Use "HMIObjects(Index)" to return an object from the HMIObjects listing, where "Index" in this case identifies the object by name: In this example the X coordinate of the OLE Element "Wordpad Document" is set to 140:

```vba
Sub EditOLEObject()
    'VBA299
    Dim objOleObject As HMIOLEObject
    Set objOleObject = ActiveDocument.HMIObjects("Wordpad Document")
    objOleObject.Left = 140
End Sub
```

Use "Selection(Index)" to return an object from the Selection listing. "For Index you can use either the index number or the name of the object. In this example the name of the first selected object will be output:

```vba
Sub ShowNameOfFirstSelectedObject()
    'VBA300
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name of the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

- How to Create Picture-specific Menus and Toolbars (Page 1655)
- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- Delete Method (Page 1831)
- AddOLEObject Method (Page 1803)
- How to Create an Application-specific Toolbar (Page 1635)
- VBA Reference (Page 1745)
- OLE Objects (Page 1669)
- Application Property (Page 2121)
- Events Property (Page 2223)
- GroupParent Property (Page 2268)
- Height Property (Page 2269)
- Layer Property (Page 2293)
- LDTooltipTexts Property (Page 2303)
- Left Property (Page 2305)
- ObjectName Property (Page 2347)
OptionGroup Object

Description

Represents the "Radio Box" object. The OptionGroup object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIOptionGroup
Usage

Use the Add method to create a new "Option Group" object in a picture:

Sub AddOptionGroup()
'VBA301
Dim objOptionGroup As HMIOptionGroup
Set objOptionGroup = ActiveDocument.HMIObjects.AddHMIObject("Radio-Box", "HMIOptionGroup")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditOptionGroup()
'VBA302
Dim objOptionGroup As HMIOptionGroup
Set objOptionGroup = ActiveDocument.HMIObjects("Radio-Box")
objOptionGroup.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA303
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

- Left Property (Page 2305)
- BorderStyle Property (Page 2160)
- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- HMIDefaultObjects Object (Listing) (Page 1985)
- AddHMIObject Method (Page 1801)
- VBA Reference (Page 1745)
- Editing Objects with VBA (Page 1660)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Top Property (Page 2446)
- ToolTipText Property (Page 2445)
Text Property (Page 2439)
Process Property (Page 2383)
PasswordLevel Property (Page 2362)
Orientation Property (Page 2356)
OperationMessage Property (Page 2354)
Operation Property (Page 2353)
Layer Property (Page 2293)
Index Property (Page 2276)
Height Property (Page 2269)
ForeFlashColorOn Property (Page 2262)
ForeFlashColorOff Property (Page 2261)
ForeColor Property (Page 2260)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FontBold Property (Page 2256)
FlashRateForeColor Property (Page 2251)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashForeColor Property (Page 2242)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BoxCount Property (Page 2164)
BoxAlignment Property (Page 2163)
BorderWidth Property (Page 2161)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
AdaptBorder Property (Page 2111)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDFonts Property (Page 2298)
LDTexts Property (Page 2303)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
WindowsStyle property (Page 2557)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
**PictureWindow Object**

**Description**

Represents the "Picture Window" object. The PictureWindow object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

HMIPictureWindow

**Usage**

Use the Add method to create a new "Picture Window" object in a picture:

```vba
Sub AddPictureWindow()
    'VBA304
    Dim objPictureWindow As HMIPictureWindow
    Set objPictureWindow = ActiveDocument.HMIObjects.AddHMIObject("PictureWindow1", "HMIPictureWindow")
    End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditPictureWindow()
    'VBA305
    Dim objPictureWindow As HMIPictureWindow
    Set objPictureWindow = ActiveDocument.HMIObjects("PictureWindow1")
    objPictureWindow.Sizeable = True
    End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA306
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also
MaximizeButton Property (Page 2322)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Zoom Property (Page 2559)
WindowBorder Property (Page 2555)
Width Property (Page 2553)
Visible Property (Page 2551)
UpdateCycle Property (Page 2463)
Top Property (Page 2446)
TagPrefix Property (Page 2436)
Sizeable Property (Page 2421)
ServerPrefix Property (Page 2416)
GetPositionY Property (Page 2409)
GetPositionX Property (Page 2408)
ScrollBars Property (Page 2408)
PictureName Property (Page 2373)
OnTop Property (Page 2352)
OffsetTop Property (Page 2351)
OffsetLeft Property (Page 2350)
Moveable Property (Page 2342)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
CloseButton Property (Page 2182)
CaptionText Property (Page 2169)
Caption Property (Page 2168)
AdaptSize Property (Page 2112)
AdaptPicture Property (Page 2111)
Application Property (Page 2121)
Events Property (Page 2223)
GroupParent Property (Page 2268)
IndependentWindow property (Page 2276)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Properties Property (Page 2388)
Selected Property (Page 2412)
ToolTipText Property (Page 2445)
Type Property (Page 2450)
WindowMonitorNumber property (Page 2556)
WindowPositionMode property (Page 2556)
ConnectionPoints property (Page 2198)
MenuToolBarConfig Property (Page 2339)
TitleBackColorActiveEnd property (Page 2440)
TitleBackColorActiveStart property (Page 2440)
TitleBackColorInactiveEnd property (Page 2440)
TitleBackColorInactiveStart property (Page 2440)
TitleForeColorActive property (Page 2441)
TitleForeColorInactive property (Page 2441)
Pinnable property (Page 2378)
Pinned property (Page 2378)
ConnectorObjects property (Page 2199)
PieSegment Object

Description

Represents the "Pie Segment" object. The PieSegment object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMI_DefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIPieSegment

Usage

Use the Add method to create a new "Pie Segment" object in a picture:

Sub AddPieSegment()
  'VBA307
  Dim objPieSegment As HMIPieSegment
  Set objPieSegment = ActiveDocument.HMIObjects.AddHMIObject("PieSegment1", "HMIPieSegment")
End Sub

Use "HMIObjects"(Index) to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditPieSegment()
  'VBA308
  Dim objPieSegment As HMIPieSegment
  Set objPieSegment = ActiveDocument.HMIObjects("PieSegment1")
  objPieSegment.BorderColor = RGB(255, 0, 0)
End Sub
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA309
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

Filling Property (Page 2232)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
StartAngle Property (Page 2426)
Radius Property (Page 2394)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
FillColor Property (Page 2231)
EndAngle Property (Page 2222)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
Polygon Object

Description

Multiple Objects

- HMIObjects
- Selection
- HMIHDefaultObjects

Represents the "Polygon" object. The Polygon object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIHDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIPolygon

Usage

Use the Add method to create a new "Polygon" object in a picture:

```vba
Sub AddPolygon()
    'VBA310
    Dim objPolygon As HMIPolygon
    Set objPolygon = ActiveDocument.HMIHObjects.AddHMIHObject("Polygon", "HMIPolygon")
End Sub
```

Use "HMIHObjects"(Index)" to return an object from the HMIHObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditPolygon()
    'VBA311
    Dim objPolygon As HMIPolygon
    Set objPolygon = ActiveDocument.HMIHObjects("Polygon")
    objPolygon.BorderColor = RGB (255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA312
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ToolTipText Property (Page 2445)
BorderBackColor Property (Page 2152)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
RotationAngle Property (Page 2400)
ReferenceRotationTop Property (Page 2397)
ReferenceRotationLeft Property (Page 2396)
PointCount Property (Page 2379)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Index Property (Page 2276)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
ActualPointTop Property (Page 2110)
ActualPointLeft Property (Page 2109)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTOoltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
PolyLine Object

Description

Represents the "Polyline" object. The PolyLine object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIPolyLine

Usage

Use the Add method to create a new "Polyline" object in a picture:

```vba
Sub AddPolyLine()
  'VBA313
  Dim objPolyLine As HMIPolyLine
  Set objPolyLine = ActiveDocument.HMIObjects.AddHMIObject("PolyLin1", "HMIPolyLine")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditPolyLine()
  'VBA314
  Dim objPolyLine As HMIPolyLine
  Set objPolyLine = ActiveDocument.HMIObjects("PolyLin1")
  objPolyLine.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA315
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name of the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

Use the "HMIObjects(Index)" to return an object from the HMIObjects Listing:

Sub EditDefaultPropertiesOfPolyLine()
    'VBA316
    Dim objPolyLine As HMIObject
    Set objPolyLine = Application.DefaultHMIObjects("HMIObject")
    objPolyLine.BorderColor = RGB(255, 255, 0)
End Sub

See also

HMIObjects Object (Listing) (Page 1985)
BorderStyle Property (Page 2156)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
RotationAngle Property (Page 2400)
ReferenceRotationTop Property (Page 2397)
ReferenceRotationLeft Property (Page 2396)
PointCount Property (Page 2379)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Index Property (Page 2276)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashBorderColor Property (Page 2240)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
ActualPointTop Property (Page 2110)
ActualPointLeft Property (Page 2109)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
Properties Object (Listing)

Description

A listing of the Property objects that represent all the properties of an object.

VBA Object Name

HMIProperties

Usage

Use the Properties(Index) property in order to return a Property object if you cannot access an object property directly. For "Index" you can use either the index number or the VBA property name of the object. In the following example the Properties property has to be used to access the individual properties of a circle. The circle will be inserted into the picture as an HMIOBJECT object:

Sub AddObject()
  'VBA319
  Dim objObject As HMIOBJECT
  Set objObject = ActiveDocument.HMIOBJECTs.AddHMIOBJECT("CircleAsHMIOBJECT", "HMICircle")
  ' Standard properties (e.g. "Position") are available every time:
  objObject.Top = 40
  objObject.Left = 40
  ' Individual properties have to be called using
  ' property "Properties":
  objObject.Properties("FlashBackColor") = True
  End Sub

See also

VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Property Object

Description

Represents the property of an object. In the case of the Property object the use of the Value property is set as the default. For this reason you can use the following notation in order for example to assign a new value to an object property:

\(<Object>\.<Property> = <Value>\)

You can use the "Dynamic" property in order to make an object property dynamic with VBA. Use the "Events" listing in order to configure actions with VBA. The Property object is an element of the Properties listing.

VBA Object Name

HMIPROPERTY

Usage

Use Properties(Index) to return an individual Property object. For "Index" you can use either the index number or the name of the object property. In the following example the name of the first property of the Circle object will be output:

Sub ShowFirstObjectOfCollection()
  'VBA317
  Dim objCircle As HMICircle
  Dim strName As String
  Set objCircle = ActiveDocument.HMIOBJECTS.AddHMIOBJECT("Circle", "HMICircle")
  strName = objCircle.Properties(1).Name
  MsgBox strName
End Sub
Use the CreateDynamic method to make an object property dynamic. In the following example the "Radius" property of a circle object will be made dynamic with the aid of the tag "Otto", which is updated every two seconds:

Sub DynamicToRadiusOfNewCircle()
'VBA318
Dim objVariableTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects("Circle")
Set objVariableTrigger = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "NewDynamic1")
objVariableTrigger.CycleType = hmiCycleType_2s
End Sub

See also

DisplayName Property (Page 2214)
Properties Object (Listing) (Page 2041)
DeleteDynamic Method (Page 1833)
CreateDynamic Method (Page 1828)
VBA Reference (Page 1745)
Creating Dynamics with VBA (Page 1686)
Editing Objects with VBA (Page 1660)
Value Property (Page 2469)
Parent Property (Page 2359)
Name Property (Page 2343)
IsDynamicable Property (Page 2280)
Events Property (Page 2223)
Dynamic Property (Page 2218)
Application Property (Page 2121)
IsPublished property (Page 2281)
QualityCodeStateValue Object

Description

QualityCodeStateValue represents the quality code of a tag which is assigned in the dynamic dialog and used for dynamization.

VBA Object Name

HMIQualityCodeStateValue

Object properties

The object QualityCodeStateValue has the following properties:

- Application
- Parent
- VALUE_BAD_COMMLUV
- VALUE_BAD_COMMLUV
- VALUE_BAD_CONFERROR
- VALUE_BAD_DEVICE
- VALUE_BAD_MISCSTATES
- VALUE_BAD_NONSPECIFIC
- VALUE_BAD_NOTCONNECTED
- VALUE_BAD_OUTOFSEVR
- VALUE_BAD_PROCRELNOM
- VALUE_BAD_PROCRELSUB
- VALUE_HIGHLIMITED
- VALUE_LOWLIMITED
- VALUE_UNCERT_ENGVHIGHLIM
- VALUE_UNCERT_ENGVLOWLIM
- VALUE_UNCERT_ENGVONLIM
- VALUE_UNCERT_INITVAL
- VALUE_UNCERT_LUV
See also

VALUE_BAD_CONFERROR Property (Page 2478)
VBA Reference (Page 1745)
VarName Property (Page 2548)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValues Object (Listing)

Description

A listing of QualityCodeStateValue objects which contain all quality codes in Dynamic dialog and are used for dynamization.

VBA Object Name

HMIQualityCodeStateValues
Application

For example, use the Item property to define values in Dynamic dialog which will be used for
dynamization when the specified tag returns the configured quality code. In the following
example the radius of a circle is given dynamics with the dynamic dialog. Dynamization uses
the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute
value (ElseCase property) is defined:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA813
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  'Activate qualitycode-statecheck
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  'define a value for every state:
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub

Object properties

The object QualityCodeStateValues has the following properties:

- Application
- Count
See also

- Item
- Parent

VALUE_UNCERT_MAINTDEM Property (Page 2532)
VBA Reference (Page 1745)
VarName Property (Page 2548)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
Parent Property (Page 2359)
Item Property (Page 2283)
Count Property (Page 2201)
Application Property (Page 2121)
Rectangle Object

Description

Represents the "Rectangle" object. The Rectangle object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIRectangle

Usage

Use the Add method to create a new "Rectangle" object in a picture:

```vba
Sub AddRectangle()
'VBA320
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditRectangle()
'VBA321
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects("Rectangle1")
objRectangle.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA322
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name of the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

- SelectedObjects object (Listing) (Page 2060)
- HMIOObjects Object (Listing) (Page 1990)
- HMIDefaultObjects Object (Listing) (Page 1985)
- AddHMIOObject Method (Page 1801)
- VBA Reference (Page 1745)
- Editing Objects with VBA (Page 1660)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Top Property (Page 2446)
- ToolTipText Property (Page 2445)
- PasswordLevel Property (Page 2362)
- Operation Property (Page 2353)
- Left Property (Page 2305)
- Layer Property (Page 2293)
- Height Property (Page 2269)
- FlashRateBorderColor Property (Page 2249)
- FlashRateBackColor Property (Page 2247)
- FlashBorderColor Property (Page 2240)
- FlashBackColor Property (Page 2239)
- FillStyle Property (Page 2235)
- FillingIndex Property (Page 2233)
- Filling Property (Page 2232)
- FillColor Property (Page 2231)
- BorderWidth Property (Page 2161)
- BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
DrawInsideFrame property (Page 2218)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)
RoundButton Object

Description

Multiple Objects

- HMIDefaultObjects
- HMIObjects
- Selection
- RoundButton

Represents the "Round Button" object. The RoundButton object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIRoundButton

Usage

Use the Add method to create a new "Round Button" object in a picture:

```vba
Sub AddRoundButton()
'VBA323
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("Roundbutton1", "HMIRoundButton")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditRoundButton()
'VBA324
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects("Roundbutton1")
ojbRoundButton.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA325
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ToolTipText Property (Page 2445)
FlashBackColor Property (Page 2239)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIOBJECT Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
Toggle Property (Page 2441)
Radius Property (Page 2394)
Pressed Property (Page 2382)
PicUpUseTransColor Property (Page 2377)
PicUpTransparent Property (Page 2376)
PicUpReferenced Property (Page 2375)
PictureUp Property (Page 2374)
PictureDown Property (Page 2372)
PictureDeactivated Property (Page 2371)
PicDownUseTransColor Property (Page 2369)
PicDownTransparent Property (Page 2368)
PicDownReferenced Property (Page 2367)
PicDeactUseTransColor Property (Page 2367)
PicDeactTransparent Property (Page 2366)
PicDeactReferenced-Eigenschaft (Page 2365)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColorTop Property (Page 2155)
BorderColor Property (Page 2153)
BorderColorBottom Property (Page 2155)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
BackBorderWidth Property (Page 2131)
AlignmentLeft Property (Page 2117)
AlignmentTop Property (Page 2118)
Application Property (Page 2121)
DisplayOptions Property (Page 2215)
Events Property (Page 2223)
FontBold Property (Page 2256)
FontItalic Property (Page 2257)
FontName Property (Page 2257)
FontSize Property (Page 2258)
FontUnderline Property (Page 2259)
RoundRectangle Object

Description
Represents the "Rounded Rectangle" object. The RoundRectangle object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

HMIRoundRectangle

**Usage**

Use the Add method to create a new "Rounded Rectangle" object in a picture:

```vba
Sub AddRoundRectangle()
'VBA326
Dim objRoundRectangle As HMIRoundRectangle
Set objRoundRectangle = ActiveDocument.HMIObjects.AddHMIObject("Roundrectangle1", "HMIRoundRectangle")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditRoundRectangle()
'VBA327
Dim objRoundRectangle As HMIRoundRectangle
Set objRoundRectangle = ActiveDocument.HMIObjects("Roundrectangle1")
objRoundRectangle.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index)" to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
'VBA328
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'SGet the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

**See also**

- Width Property (Page 2553)
- BorderBackColor Property (Page 2152)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
RoundCornerWidth Property (Page 2402)
RoundCornerHeight Property (Page 2401)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
Application Property (Page 2121)
Events Property (Page 2218)
GlobalColorScheme property (Page 2264)
ScriptInfo Object

Description

Represented as a script (C, VB) that is configured for adding dynamics to a property or action to an event.

VBA Object Name

HMIScriptInfo
Usage

Use the CreateDynamic method to make a property dynamic with the aid of a script. In the following example...

Sub AddDynamicAsCSkriptToProperty()
"VBA329
Dim objCScript As HMIScriptInfo
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")
Set objCScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeCScript)
' Define triggertype and cycletime:
With objCScript
 .SourceCode = ""
 .Trigger.Type = hmiTriggerTypeStandardCycle
 .Trigger.CycleType = hmiCycleType_2s
 .Trigger.Name = "Trigger1"
End With
End Sub

Use the AddAction method to configure an action on an event. In the following example...

Sub AddActionToPropertyTypeCScript()
"VBA330
Dim objEvent As HMIEvent
Dim objCScript As HMIScriptInfo
Dim objCircle As HMICircle
' Add circle to picture. By changing of property "Radius"
' a C-action is added:
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
Set objEvent = objCircle.Radius.Events(1)
Set objCScript = objEvent.Actions.AddAction(hmiActionCreationTypeCScript)
End Sub

See also

Prototype Property (Page 2389)
Delete Method (Page 1831)
VBA Reference (Page 1745)
Creating Dynamics with VBA (Page 1686)
Trigger Property (Page 2449)
SourceCode Property (Page 2424)
ScriptType Property (Page 2407)
Parent Property (Page 2359)
Compiled Property (Page 2197)
SelectedObjects object (Listing)

Description

A listing of the HMIObject objects that represent all the selected objects in a picture.

VBA Object Name

HMISelectedObjects

Usage

Use the Selection property to return the Selection listing. In the following example the names of all the selected objects in the active picture will be output:

Sub ShowSelectionOfDocument()
    'VBA331
    Dim colSelection As HMISelectedObjects
    Dim objObject As HMIObject
    Dim strObjectList As String
    Set colSelection = ActiveDocument.Selection
    If colSelection.Count <> 0 Then
        strObjectList = "List of selected objects:"
        For Each objObject In colSelection
            strObjectList = strObjectList & vbCrLf & objObject.ObjectName
        Next objObject
    Else
        strObjectList = "No objects selected"
    End If
    MsgBox strObjectList
End Sub
Use the SelectAll method, for example, to select all the objects in the picture. In the following example all the objects in the active picture are selected:

Sub SelectAllObjects()
    'VBA332
    ActiveDocument.Selection.SelectAll
End Sub

See also

HMIObjects Object (Listing) (Page 1990)
AlignTop Method (Page 1810)
HMIDefaultObjects Object (Listing) (Page 1985)
BringToFront Method (Page 1813)
SendToBack Method (Page 1883)
SelectAll Method (Page 1882)
SameWidthAndHeight Method (Page 1877)
SameWidth Method (Page 1876)
SameHeight Method (Page 1874)
Rotate Method (Page 1872)
Remove Method (Page 1871)
ForwardOneLevel Method (Page 1846)
BackwardOneLevel Method (Page 1811)
MoveSelection Method (Page 1865)
Item Method (Page 1859)
FlipVertically Method (Page 1845)
FlipHorizontally Method (Page 1843)
EvenlySpaceVertically Method (Page 1838)
EvenlySpaceHorizontally Method (Page 1837)
DuplicateSelection Method (Page 1836)
DeselectAll Method (Page 1834)
DeleteAll Method (Page 1832)
CreateGroup Method (Page 1829)
CreateCustomizedObject Method (Page 1825)
CopySelection Method (Page 1823)
CenterVertically Method (Page 1816)
Slider object

Description

Represented by the object called **Slider Object**. The **Slider object** is an element of the following listings:

- **Objects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

**HMISlider**
Usage

Use the Add method to create a new "Slider Object" object in a picture:

```vba
Sub AddSlider()
'VBA333
Dim objSlider As HMISlider
Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("Slider1", "HMISlider")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditSlider()
'VBA334
Dim objSlider As HMISlider
Set objSlider = ActiveDocument.HMIObjects("Slider1")
objSlider.ButtonColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
'VBA335
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

- OperationReport Property (Page 2355)
- BorderFlashColorOff Property (Page 2157)
- SelectedObjects object (Listing) (Page 2060)
- HMIObjects Object (Listing) (Page 1990)
- HMIDefaultObjects Object (Listing) (Page 1985)
- AddHMIObject Method (Page 1801)
- VBA Reference (Page 1745)
- Editing Objects with VBA (Page 1660)
- Width Property (Page 2553)
- Visible Property (Page 2551)
- Top Property (Page 2446)
- ToolTipText Property (Page 2445)
SmallChange Property (Page 2421)
Process Property (Page 2383)
PasswordLevel Property (Page 2362)
OperationMessage Property (Page 2354)
Operation Property (Page 2353)
Min Property (Page 2340)
Max Property (Page 2322)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
ExtendedOperation Property (Page 2227)
Direction Property (Page 2213)
ColorTop Property (Page 2191)
ColorBottom Property (Page 2184)
ButtonColor Property (Page 2167)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColorTop Property (Page 2135)
BackColor Property (Page 2131)
BackColorBottom Property (Page 2135)
BackBorderWidth Property (Page 2131)
Application Property (Page 2121)
SourceLink Object

Description

```
  DirectConnection
    DesiLink
    SourceLink
```

Represents the source for a direct connection.

VBA Object Name

HMISourceLink
Usage

Use the SourceLink property to return the SourceLink object. In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
  'VBA336
  Dim objButton As HMIButton
  Dim objRectangleA As HMIRectangle
  Dim objRectangleB As HMIRectangle
  Dim objEvent As HMIEvent
  Dim objDirConnection As HMIDirectConnection

  'Add objects to active document:
  Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
  Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
  With objRectangleA
    .Top = 100
    .Left = 100
  End With
  With objRectangleB
    .Top = 250
    .Left = 400
    .BackColor = RGB(255, 0, 0)
  End With
  With objButton
    .Top = 10
    .Left = 10
    .Text = "SetPosition"
  End With

  'Initiation of directconnection by mouseclick:
  Set objDirConnection =
    objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
  With objDirConnection
    'Sourceobject: Top-property of Rectangle_A
    .SourceLink.Type = hmiSourceTypeProperty
    .SourceLink.ObjectName = "Rectangle_A"
    .SourceLink.AutomationName = "Top"
    ,
    'Targetobject: Left-property of Rectangle_B
    .DestinationLink.Type = hmiDestTypeProperty
    .DestinationLink.ObjectName = "Rectangle_B"
    .DestinationLink.AutomationName = "Left"
  End With
End Sub
See also

DirectConnection Object (Page 1947)
VBA Reference (Page 1745)
Type Property (Page 2450)
SourceLink Property (Page 2422)
ObjectName Property (Page 2347)
AutomationName Property (Page 2124)
Application Property (Page 2121)
Parent Property (Page 2359)

StaticText Object

Description

Multiple Objects

HMIObject

Selection

HMIDefaultObjects

StaticText

Represents the "Static Text" object. The StaticText object is an element of the following listings:
- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMIStaticText
Usage

Use the Add method to create a new "Static Text" object in a picture:

Sub AddStaticText()
'VBA337
Dim objStaticText As HMIStaticText
Set objStaticText = ActiveDocument.HMIObjects.AddHMIObject("Static_Text1", "HMIStaticText")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditStaticText()
'VBA338
Dim objStaticText As HMIStaticText
Set objStaticText = ActiveDocument.HMIObjects("Static_Text1")
objStaticText.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA339
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

SelectedObjects object (Listing) (Page 2060)
FontBold Property (Page 2256)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
Text Property (Page 2439)
PasswordLevel Property (Page 2362)
Orientation Property (Page 2356)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Height Property (Page 2269)
ForeFlashColorOn Property (Page 2262)
ForeFlashColorOff Property (Page 2261)
ForeColor Property (Page 2260)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FlashRateForeColor Property (Page 2251)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashForeColor Property (Page 2242)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillingIndex Property (Page 2233)
Filling Property (Page 2232)
FillColor Property (Page 2231)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
AdaptBorder Property (Page 2111)
StatusDisplay Object

Description

Multiple Objects

HMIUserObjects
HMIObjects
Selection

StatusDisplay
Represents the "Status Display" object. The "StatusDisplay" object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

**VBA Object Name**

HMIStatusDisplay

**Usage**

Use the Add method to create a new "Status Display" object in a picture:

```vba
Sub AddStatusDisplay()
  'VBA340
  Dim objStatusDisplay As HMIStatusDisplay
  Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("Statusdisplay1", 
  "HMIStatusDisplay")
End Sub
```

Use "HMIObjects(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditStatusDisplay()
  'VBA341
  Dim objStatusDisplay As HMIStatusDisplay
  Set objStatusDisplay = ActiveDocument.HMIObjects("Statusdisplay1")
  objStatusDisplay.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection(Index)" to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
  'VBA342
  'Select all objects in the picture:
  ActiveDocument.Selection.SelectAll
  'Get the name of the first object of the selection:
  MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

**See also**

- ToolTipText Property (Page 2445)
- BasePicReferenced Property (Page 2143)
SelectedObjects object (Listing) (Page 2060)
HMIOBJECTS Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Width Property (Page 2553)
Visible Property (Page 2551)
Top Property (Page 2446)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Left Property (Page 2305)
Layer Property (Page 2293)
Index Property (Page 2276)
Height Property (Page 2269)
FlashRateFlashPic Property (Page 2250)
FlashRateBorderColor Property (Page 2249)
FlashPicUseTransColor Property (Page 2246)
FlashPicture Property (Page 2244)
FlashPicTransColor Property (Page 2243)
FlashPicReferenced Property (Page 2242)
FlashFlashPicture Property (Page 2241)
FlashBorderColor Property (Page 2240)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BasePicUseTransColor Property (Page 2145)
BasePicture Property (Page 2144)
BasePicTransColor Property (Page 2144)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
SymbolLibraries Object (Listing)

Description

![Diagram of SymbolLibraries object structure]

A listing of the SymbolLibrary objects that represent the Components Library. The listing contains two objects: The first object is the "Global Library" and the second object is the "Project Library".

VBA Object Name

HMISymbolLibraries
Usage

Use the SymbolLibraries property to return the SymbolLibraries listing. In the following example the names of the libraries will be output:

Sub ShowSymbolLibraries()
    'VBA344
    Dim colSymbolLibraries As HMISymbolLibraries
    Dim objSymbolLibrary As HMISymbolLibrary
    Dim strLibraryList As String
    Set colSymbolLibraries = Application.SymbolLibraries
    For Each objSymbolLibrary In colSymbolLibraries
        strLibraryList = strLibraryList & objSymbolLibrary.Name & vbCrLf
    Next objSymbolLibrary
    MsgBox strLibraryList
End Sub

See also

SymbolLibrary Object (Page 2074)
Item Method (Page 1859)
VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)
Parent Property (Page 2359)
Count Property (Page 2201)
Application Property (Page 2121)

SymbolLibrary Object

Description

Represents the "Global Library" or "Project Library". The SymbolLibrary object is an element of the SymbolLibraries listing.
VBA Object Name

HMISymbolLibrary

Usage

Use SymbolLibraries(Index) to return an individual SymbolLibrary object. "For Index you can use either the index number or the name of the object. In the following example the name of the "Global Library" will be output:

```vba
Sub ShowFirstObjectOfCollection()
    'VBA343
    Dim strName As String
    strName = Application.SymbolLibraries(1).Name
    MsgBox strName
End Sub
```

See also

SymbolLibraries Object (Listing) (Page 2073)
GetItemByPath Method (Page 1847)
FindByDisplayName Method (Page 1842)
VBA Reference (Page 1745)
Accessing the component library with VBA (Page 1646)
Parent Property (Page 2359)
Name Property (Page 2343)
FolderItems Property (Page 2254)
Application Property (Page 2121)
TextList Object

Description

Represents the "Text List" object. The TextList object is an element of the following listings:

- Objects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all standard, Windows, and smart objects.

VBA Object Name

HMITextList

Usage

Use the Add method to create a new "Text List" object in a picture:

```vba
Sub AddTextList()
    'VBA345
    Dim objTextList As HMITextList
    Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("Textlist1", "HMITextList")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditTextList()
    'VBA346
    Dim objTextList As HMITextList
    Set objTextList = ActiveDocument.HMIObjects("Textlist1")
    objTextList.BorderColor = RGB(255, 0, 0)
End Sub
```
Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA347
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

Width Property (Page 2553)
ForeFlashColorOn Property (Page 2262)
BitNumber Property (Page 2148)
SelectedObjects object (Listing) (Page 2060)
HMIObjects Object (Listing) (Page 1990)
HMIDefaultObjects Object (Listing) (Page 1985)
AddHMIObject Method (Page 1801)
VBA Reference (Page 1745)
Editing Objects with VBA (Page 1660)
Visible Property (Page 2551)
UnselTextColor Property (Page 2462)
UnselBGColor Property (Page 2462)
Top Property (Page 2446)
ToolTipText Property (Page 2445)
SelTextColor Property (Page 2414)
SelBGColor Property (Page 2412)
PasswordLevel Property (Page 2362)
OutputValue Property (Page 2358)
Orientation Property (Page 2356)
OperationReport Property (Page 2355)
OperationMessage Property (Page 2354)
Operation Property (Page 2353)
NumberLines Property (Page 2346)
ListType Property (Page 2312)
Left Property (Page 2305)
Layer Property (Page 2293)
LanguageSwitch Property (Page 2291)
ItemBorderWidth Property (Page 2286)
ItemBorderStyle Property (Page 2285)
ItemBorderColor Property (Page 2284)
ItemBorderBackColor Property (Page 2283)
Height Property (Page 2269)
ForeFlashColorOff Property (Page 2261)
ForeColor Property (Page 2260)
FontUnderline Property (Page 2259)
FontSize Property (Page 2258)
FontName Property (Page 2257)
FontItalic Property (Page 2257)
FontBold Property (Page 2256)
FlashRateForeColor Property (Page 2251)
FlashRateBorderColor Property (Page 2249)
FlashRateBackColor Property (Page 2247)
FlashForeColor Property (Page 2242)
FlashBorderColor Property (Page 2240)
FlashBackColor Property (Page 2239)
FillStyle Property (Page 2235)
FillColor Property (Page 2231)
EditAtOnce Property (Page 2219)
CursorControl Property (Page 2206)
BoxType Property (Page 2165)
BorderWidth Property (Page 2161)
BorderStyle Property (Page 2160)
BorderFlashColorOn Property (Page 2159)
BorderFlashColorOff Property (Page 2157)
BorderColor Property (Page 2153)
BorderBackColor Property (Page 2152)
BackFlashColorOn Property (Page 2139)
BackFlashColorOff Property (Page 2138)
BackColor Property (Page 2131)
AssumeOnExit Property (Page 2123)
Assignments Property (Page 2122)
AlignmentTop Property (Page 2118)
AlignmentLeft Property (Page 2117)
AdaptBorder Property (Page 2111)
Application Property (Page 2121)
Events Property (Page 2223)
GlobalColorScheme property (Page 2264)
GlobalShadow property (Page 2264)
GroupParent Property (Page 2268)
InputValue property (Page 2279)
LDTooltipTexts Property (Page 2303)
ObjectName Property (Page 2347)
Parent Property (Page 2359)
Properties Property (Page 2388)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Transparency property (Page 2447)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
DropDownListStyle property (Page 2218)
LDAPAssignments property (Page 2298)
TextBiblIDs property (Page 2439)
ConnectorObjects property (Page 2199)
**Toolbar Object**

**Description**

Represents the "User Defined Toolbar" object. The Toolbar object is an element of the CustomToolbars listing.

**VBA Object Name**

HMIToolbar

**Usage**

Use CustomToolbars(Index) to return an individual Toolbar object. "For Index you can use either the index number or the name of the object. In the following example the "Key" parameter of the first user-defined toolbar in the active picture will be output:

```vba
Sub ShowFirstObjectOfCollection()
  'VBA348
  Dim strName As String
  strName = ActiveDocument.CustomToolbars(1).Key
  MsgBox strName
End Sub
```

Use the Delete method to remove a "Toolbar" object from the "CustomToolbars" listing. In the following example the first user-defined toolbar in the active picture will be removed:

```vba
Sub DeleteToolbar()
  'VBA349
  Dim objToolbar As HMIToolbar
  Set objToolbar = ActiveDocument.CustomToolbars(1)
  objToolbar.Delete
End Sub
```
See also

- Key Property (Page 2287)
- Toolbars Object (Listing) (Page 2081)
- Delete Method (Page 1831)
- How to Create Picture-specific Menus and Toolbars (Page 1655)
- How to Create an Application-specific Toolbar (Page 1635)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
- Visible Property (Page 2551)
- ToolbartItems Property (Page 2443)
- Parent Property (Page 2359)
- Application Property (Page 2121)

**Toolbars Object (Listing)**

**Description**

A listing of the Toolbar objects that represent all the user-defined toolbars in the Graphics Designer.

**VBA Object Name**

HMI\-Custom\-Toolbars

**Usage**

**Note**

In order for the examples to work, first create a user-defined toolbar. For an example of this, please refer to "Creating a New Application-Specific Toolbar" in this documentation.
Use the CustomToolbars property to return the Toolbars listing. In the following example, values for the "Key" property of all user-defined toolbars in the active picture will be output:

**Note**

The Toolbars listing does not distinguish between application-specific and picture-specific toolbars in the output.

Sub ShowCustomToolbarsOfDocument()
'VBA350
Dim colToolbars As HMIToolbars
Dim objToolbar As HMIToolbar
Dim strToolbarList As String
Set colToolbars = ActiveDocument.CustomToolbars
If 0 <> colToolbars.Count Then
    For Each objToolbar In colToolbars
        strToolbarList = strToolbarList & objToolbar.Key & vbCrLf
    Next objToolbar
Else
    strToolbarList = "No toolbars existing"
End If
MsgBox strToolbarList
End Sub

Use the Application property and the Add method if you want to create an application-specific toolbar. Create the VBA code in either the "Project Template" document or the "Global Template" document.

Sub InsertApplicationSpecificToolbar()
'VBA351
Dim objToolbar As HMIToolbar
Set objToolbar = Application.CustomToolbars.Add("a_Toolbar1")
End Sub

Use the ActiveDocument property and the Add method if you want to create a picture-specific toolbar. Create the VBA code in the document called "ThisDocument":

Sub InsertDocumentSpecificToolbar()
'VBA352
Dim objToolbar As HMIToolbar
Set objToolbar = ActiveDocument.CustomToolbars.Add("d_Toolbar1")
End Sub

See also

Toolbar Object (Page 2080)
Item Method (Page 1859)
ToolbarItem Object

Description

 Represents an object (icon or dividing line) in a user-defined toolbar in the GraphicsDesigner. The ToolbarItem object is an element of the ToolbarItems listing.

VBA Object Name

HMIToolbarItem

Usage

Note

In order for the examples to work, first create a user-defined toolbar. For an example of this, please refer to "Creating a New Application-Specific Toolbar" in this documentation.
Use `ToolbarItems(Index)` to return an individual `ToolbarItem` object. "For Index you can use either the index number or the name of the object. In the following example the type of the first object in the first user-defined toolbar in the active picture will be output:

```vba
Sub ShowFirstObjectOfCollection()
    'VBA353
    Dim strType As String
    strType = ActiveDocument.CustomToolbars(1).ToolbarItems(1).ToolbarItemType
    MsgBox strType
End Sub
```

Use the `Delete` method to remove an object from the "`ToolbarItems" listing. In the following example the first object will be deleted from the first user-defined toolbar in the active picture:

```vba
Sub DeleteToolbarItem()
    'VBA354
    ActiveDocument.CustomToolbars(1).ToolbarItems(1).Delete
End Sub
```

See also

- Macro Property (Page 2320)
- `ToolbarItems` Object (Listing) (Page 2085)
- Delete Method (Page 1831)
- Configuring Menus and Toolbars (Page 1626)
- How to assign VBA macros to menus and toolbars (Page 1642)
- How to assign help texts to menus and toolbars (Page 1639)
- How to Add a New Icon to the Toolbar (Page 1637)
- VBA Reference (Page 1745)
- Creating Customized Menus and Toolbars (Page 1627)
- Visible Property (Page 2551)
- Type Property (Page 2450)
- ToolTipText Property (Page 2445)
- Tag Property (Page 2434)
-(StatusText Property (Page 2426)
- ShortCut Property (Page 2416)
- Position Property (Page 2380)
- Parent Property (Page 2359)
- LDTooltipTexts Property (Page 2303)
- LDStatusTexts Property (Page 2301)
ToolbarItems Object (Listing)

Description

A listing of the ToolbarItem objects that represent all the objects in a user-defined toolbar.

VBA Object Name

HMIToolbarItems

Usage

Use the ToolbarItems property to return the ToolbarItems listing. In the following example, all object types in the first user-defined toolbar in the active picture will be output:

Note

The ToolbarItems listing does not distinguish between application-specific and picture-specific toolbars in the output.
Sub ShowToolbarItems()
'VBA355
Dim colToolbarItems As HMIToolbarItems
Dim objToolbarItem As HMIToolbarItem
Dim strTypeList As String
Set colToolbarItems = ActiveDocument.CustomToolbars(1).ToolbarItems
If 0 <> colToolbarItems.Count Then
    For Each objToolbarItem In colToolbarItems
        strTypeList = strTypeList & objToolbarItem.ToolbarItemType & vbCrLf
    Next objToolbarItem
Else
    strTypeList = "No Toolbaritems existing"
End If
MsgBox strTypeList
End Sub

Use the InsertToolbarItem method, for instance, to insert an icon into an existing user-defined toolbar. In the following example a picture-specific toolbar will be created in the active picture and an icon will be added:

Sub InsertToolbarItem()
'VBA356
Dim objToolbar As HMIToolbar
Dim objToolbarItem As HMIToolbarItem
Set objToolbar = ActiveDocument.CustomToolbars.Add("d_Toolbar2")
Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "t_Item2_1", "ToolbarItem 1")
End Sub

See also

ToolbarItem Object (Page 2083)
InsertToolbarItem Method (Page 1856)
InsertSeparator Method (Page 1853)
InsertFromMenuitem Method (Page 1848)
How to Add a New Icon to the Toolbar (Page 1637)
VBA Reference (Page 1745)
Creating Customized Menus and Toolbars (Page 1627)
Parent Property (Page 2359)
Count Property (Page 2201)
Application Property (Page 2121)
Trigger Object

Description

Represents the trigger (e.g. Picture Cycle) that is necessary for adding dynamics to properties with the aid of scripts. A trigger can possess multiple tag triggers.

VBA Object Name

HMITrigger

Usage

Use the Trigger property to return the Trigger object. In this example the “Radius” property of a circle will be made dynamic with the aid of a VB script (the output value sets the radius):

```
Sub AddDynamicAsVBSkriptToProperty()
    'VBA357
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")
    Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    'Define cycletime and sourcecode
    With objVBScript
        .SourceCode = ""
        .Trigger.Type = hmiTriggerTypeStandardCycle
        .Trigger.CycleType = hmiCycleType_2s
        .Trigger.Name = "Trigger1"
    End With
End Sub
```

See also

Delete Method (Page 1831)

VBA Reference (Page 1745)
TubeArcObject object

Description

Represents the “Tube arc” object. The TubeArcObject object is an element of the following lists:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all default, smart, window and tube objects.

**VBA object name**

HMITubeArcObject
Usage

Use the Add method to create a new "Tube arc" object in a picture:

```vba
Sub AddTubeArcObject()
    'VBA835
    Dim objTubeArcObject As HMITubeArcObject
    Set objTubeArcObject = ActiveDocument.HMIObjects.AddHMIObject("TubeArcObject", "HMITubeArcObject")
    End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditTubeArcObject()
    'VBA836
    Dim objTubeArcObject As HMITubeArcObject
    Set objTubeArcObject = ActiveDocument.HMIObjects("TubeArcObject")
    objTubeArcObject.BorderColor = RGB(255, 0, 0)
    End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
    'VBA837
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
    End Sub
```

See also

- ObjectName Property (Page 2347)
- Left Property (Page 2305)
- Layer Property (Page 2293)
- Top Property (Page 2446)
- Width Property (Page 2553)
- Height Property (Page 2269)
- BorderColor Property (Page 2153)
- BorderWidth Property (Page 2161)
- ToolTipText Property (Page 2445)
- Visible Property (Page 2551)
- PasswordLevel Property (Page 2362)
TubeDoubleTeeObject object

Description

Represents the "Double T-piece" object. The TubeDoubleTeeObject object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
- HMIDefaultObjects: Contains the default property values of all default, smart, window and tube objects.

VBA object name

HMITubeDoubleTeeObject
Usage

Use the Add method to create a new "Double T-piece" object in a picture:

Sub AddTubeDoubleTeeObject()
    'VBA838
    Dim objTubeDoubleTeeObject As HMITubeDoubleTeeObject
    Set objTubeDoubleTeeObject = ActiveDocument.HMIObjects.AddHMIObject("Double T-piece",
        "HMITubeDoubleTeeObject")
    End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditTubeDoubleTeeObject()
    'VBA839
    Dim objTubeDoubleTeeObject As HMITubeDoubleTeeObject
    Set objTubeDoubleTeeObject = ActiveDocument.HMIObjects("Double T-piece")
    objTubeDoubleTeeObject.BorderColor = RGB(255, 0, 0)
    End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
    'VBA840
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
    End Sub

See also

ObjectName Property (Page 2347)
Left Property (Page 2305)
Layer Property (Page 2293)
Top Property (Page 2446)
Width Property (Page 2553)
Height Property (Page 2269)
BorderColor Property (Page 2153)
BorderWidth Property (Page 2161)
ToolTipText Property (Page 2445)
Visible Property (Page 2551)
PasswordLevel Property (Page 2362)
TubePolyline object

Description

![Diagram of TubePolyline object]

Represents the “TubePolyline” object. The TubePolyline object is an element of the following listings:

- **HMIOBJECTS**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all default, smart, window and tube objects.
VBA object name

HMITubePolyline

Usage

Use the Add method to create a new "TubePolyline" object in a picture:

Sub AddTubePolyline()
'VBA841
Dim objTubePolyline As HMITubePolyline
Set objTubePolyline = ActiveDocument.HMIObjects.AddHMIObject("TubePolyline",
"HMITubePolyline")
End Sub

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

Sub EditTubePolyline()
'VBA842
Dim objTubePolyline As HMITubePolyline
Set objTubePolyline = ActiveDocument.HMIObjects("TubePolyline")
objTubePolyline.BorderColor = RGB(255, 0, 0)
End Sub

Use "Selection"(Index) to return an object from the Selection listing:

Sub ShowNameOfFirstSelectedObject()
'VBA843
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub

See also

ObjectName Property (Page 2347)
Left Property (Page 2305)
Layer Property (Page 2293)
Top Property (Page 2446)
Width Property (Page 2553)
Height Property (Page 2269)
BorderColor Property (Page 2153)
BorderWidth Property (Page 2161)
ToolTipText Property (Page 2445)
Visible Property (Page 2551)
PasswordLevel Property (Page 2362)
Operation Property (Page 2353)
Transparency property (Page 2447)
GlobalShadow property (Page 2264)
GlobalColorScheme property (Page 2264)
PointCount Property (Page 2379)
ActualPointLeft Property (Page 2109)
ActualPointTop Property (Page 2110)
Index Property (Page 2276)
Application Property (Page 2121)
Events Property (Page 2223)
BorderStyle Property (Page 2160)
GroupParent Property (Page 2268)
LDTOoltipTexts Property (Page 2303)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
Type Property (Page 2450)
ConnectionPoints property (Page 2198)
ConnectorObjects property (Page 2199)

**TubeTeeObject object**

**Description**

![Diagram of TubeTeeObject object hierarchy]
Represents the “T-piece” object. The TubeTeeObject object is an element of the following lists:

- **HMIObjects**: Contains all objects of a picture.
- **Selection**: Contains all selected objects of a picture.
- **HMIDefaultObjects**: Contains the default property values of all default, smart, window and tube objects.

### VBA object name

HMITubeTeeObject

### Usage

Use the Add method to create a new “T-piece” object in a picture:

```vba
Sub AddTubeTeeObject()
    'VBA844
    Dim objTubeTeeObject As HMITubeTeeObject
    Set objTubeTeeObject = ActiveDocument.HMIObjects.AddHMIObject("T-piece", "HMITubeTeeObject")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```vba
Sub EditTubeTeeObject()
    'VBA845
    Dim objTubeTeeObject As HMITubeTeeObject
    Set objTubeTeeObject = ActiveDocument.HMIObjects("T-piece")
    objTubeTeeObject.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

```vba
Sub ShowNameOfFirstSelectedObject()
    'VBA846
    'Select all objects in the picture:
    ActiveDocument.Selection.SelectAll
    'Get the name from the first object of the selection:
    MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

### See also

- **ObjectName Property (Page 2347)**
- **Left Property (Page 2305)**
VariableStateValue Object

Description

```
DynamicDialogInfo

VariableStateValues

VariableStateValue
```
Represents the state of a tag, the value of which is assigned in the Dynamic dialog and used

**VBA Object Name**

HMIVariableStateValue

**See also**

- VALUE_SERVERDOWN Property (Page 2516)
- VBA Reference (Page 1745)
- VarName Property (Page 2548)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
- VALUE_MAX_RANGE Property (Page 2508)
- VALUE_MAX_LIMIT Property (Page 2506)
- VALUE_INVALID_KEY Property (Page 2502)
- VALUE_HARDWARE_ERROR Property (Page 2498)
- VALUE_HANDSHAKE_ERROR Property (Page 2496)
- VALUE_CONVERSION_ERROR Property (Page 2494)
- VALUE_ADDRESS_ERROR Property (Page 2472)
- VALUE_ACCESS_FAULT Property (Page 2470)
- Parent Property (Page 2359)
- Application Property (Page 2121)

**VariableStateValues Object (Listing)**

**Description**

A listing of VariableStateValue objects containing all tag statuses in Dynamic dialog to be used for dynamization.
**VBA Object Name**

HMIVariableStateValues

**Usage**

Use the Item property in the Dynamic dialog to define values that will be used for creating dynamics when the specified tag returns the configured state. In the following example the radius of a circle is given dynamics with the The dynamization takes place be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA358
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        'Activate variable-statecheck
        .VariableStateChecked = True
    End With
    With objDynDialog.VariableStateValues(1)
        'define a value for every state:
        .VALUE_ACCESS_FAULT = 20
        .VALUE_ADDRESS_ERROR = 30
        .VALUE_CONVERSION_ERROR = 40
        .VALUE_HANDSHAKE_ERROR = 60
        .VALUE_HARDWARE_ERROR = 70
        .VALUE_INVALID_KEY = 80
        .VALUE_MAX_LIMIT = 90
        .VALUE_MAX_RANGE = 100
        .VALUE_MIN_LIMIT = 110
        .VALUE_MIN_RANGE = 120
        .VALUE_NOT_ESTABLISHED = 130
        .VALUE_SERVERDOWN = 140
        .VALUE_STARTUP_VALUE = 150
        .VALUE_TIMEOUT = 160
    End With
End Sub
```

**See also**

- VALUE_MAX_RANGE Property (Page 2508)
- VBA Reference (Page 1745)
- VarName Property (Page 2548)
- VALUE_TIMEOUT Property (Page 2520)
VariableTrigger Object

Description

Multiple Objects

Trigger

VariableTriggers

VariableTrigger

Represents a tag trigger.

VBA Object Name

HMIVariableTrigger
Use the VariableTrigger object in order to edit or delete an existing tag trigger. In this example a circle property "Top" is made dynamic with the aid of the tag "NewDynamic1":

Sub AddDynamicAsVariableDirectToProperty()
'VBA359
Dim objVariableTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")
Set objVariableTrigger = objCircle.Top.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "'NewDynamic1'")
' Define cycletime
With objVariableTrigger
  .CycleType = hmiCycleType_2s
End With
End Sub

See also
Delete Method (Page 1831)
VariableTriggers Object (Listing) (Page 2101)
VBA Reference (Page 1745)
VariableTriggers Property (Page 2548)
Type Property (Page 2450)
Parent Property (Page 2359)
Name Property (Page 2343)
CycleType Property (Page 2209)
Application Property (Page 2121)
CycleName Property (Page 2208)
CycleTime Property (Page 2209)
VarName Property (Page 2548)
VariableTriggers Object (Listing)

Description

A listing of the VariableTrigger objects that represent all the tag triggers in use.

VBA Object Name

HMIVariableTriggers

Usage

Use the Add method to create a new tag trigger. In the following example the radius of a circle is made dynamic with the aid of a VB script. A tag trigger is used as the trigger:

```vba
Sub DynamicWithVariableTriggerCycle()
'VBA360
Dim objVBScript As HMIScriptInfo
Dim objVarTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VariableTrigger", "HMICircle")
Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
With objVBScript
'Definition of triggername and cycletime is to do with the Add-methode
Set objVarTrigger = .Trigger.VariableTriggers.Add("VarTrigger", hmiVariableCycleType_10s)
.SourceCode = ""
End With
End Sub
```

See also

Add Method (TagTriggers Listing) (Page 1792)
VBA Reference (Page 1745)
Parent Property (Page 2359)
Item Property (Page 2283)
Count Property (Page 2201)
Application Property (Page 2121)

View Object

Description

![Diagram: Document structure with Views and View]

Represents a copy of a picture. The View object is an element of the Views listing. You can use the properties of the View object among other things to control the visibility of the CS layers and to define the zoom.

VBA Object Name

HMIView

Usage

Use Views(Index) to return an individual View object. In the following example the number of copies of the active picture will be output:

```vba
Sub ShowNumberOfExistingViews()
    'VBA361
    Dim iMaxViews As Integer
    iMaxViews = ActiveDocument.Views.Count
    MsgBox "Number of copies from active document: " & iMaxViews
End Sub
```

Use the Add method to add a new View object to the "Views" listing. In the following example a copy of the active picture is created and then activated:

```vba
Sub AddView()
    'VBA362
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
End Sub
```
See also

Height Property (Page 2269)
Views Object (Listing) (Page 2103)
SetCSLayerVisible Method (Page 1885)
PrintDocument Method (Page 1869)
IsCSLayerVisible Method (Page 1857)
Delete Method (Page 1831)
Add Method (Views Listing) (Page 1793)
Activate Method (Page 1785)
VBA Reference (Page 1745)
Editing a Copy of a Picture with VBA (Page 1658)
Editing Layers with VBA (Page 1657)
ExtendedZoomingEnable Property (Page 2228)
Zoom Property (Page 2559)
WindowState Property (Page 2557)
Width Property (Page 2553)
Top Property (Page 2446)
ScrollPosY Property (Page 2411)
ScrollPosX Property (Page 2410)
Parent Property (Page 2359)
Left Property (Page 2305)
IsActive Property (Page 2279)
Application Property (Page 2121)
ActiveLayer Property (Page 2108)

Views Object (Listing)

Description

A listing of the View objects that represent a copy of a picture.
VBA Object Name

HMIViews

Usage

Use the Views listing to return a View object. In the following example the number of existing copies of the active picture will be output:

```vba
Sub ShowNumberOfExistingViews()
'VBA363
Dim iMaxViews As Integer
iMaxViews = ActiveDocument.Views.Count
MsgBox "Number of copies from active document: " & iMaxViews
End Sub
```

Use the Add method to create a copy of a picture. In the following example a copy of the active picture is created and then activated:

```vba
Sub AddViewToActiveDocument()
'VBA364
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Activate
End Sub
```

See also

- Item Method (Page 1859)
- View Object (Page 2102)
- Add Method (Page 1786)
- VBA Reference (Page 1745)
- Parent Property (Page 2359)
- Count Property (Page 2201)
- Application Property (Page 2121)

WPFControl object

Description

Represents the "WPFControl" object. The WPFControl object is an element of the following listings:

- HMIObjects: Contains all objects of a picture.
- Selection: Contains all selected objects of a picture.
VBA Object Name

HMIWPFControl

Application

Use the AddWPFControl method to insert a WPFControl in a picture.

In the following example, the "WPF Control" object outside the Global Assembly Cache is inserted in the active picture.

'VBA852
Dim WPFControl As HMIWPFControl
Set WPFControl = ActiveDocument.HMIObjects.AddWPFControl("MyWPFVBAControl", "WinCCWPFControl.TestControl", False, "Assembly=Z:\TestControl\WinCCWPFControl.dll")

See also

AddWPFControl method (Page 1806)
Delete Method (Page 1831)
Application Property (Page 2121)
AssemblyInfo property (Page 2122)
ControlType property (Page 2201)
Events Property (Page 2223)
GroupParent Property (Page 2268)
Height Property (Page 2269)
Layer Property (Page 2293)
LDTooltipTexts Property (Page 2303)
Left Property (Page 2305)
ObjectName Property (Page 2347)
Operation Property (Page 2353)
Parent Property (Page 2359)
PasswordLevel Property (Page 2362)
Properties Property (Page 2388)
Selected Property (Page 2412)
TabOrderSwitch Property (Page 2433)
TabOrderAlpha Property (Page 2429)
ToolTipText Property (Page 2445)
Top Property (Page 2446)
Type Property (Page 2450)
Visible Property (Page 2551)
Width Property (Page 2553)
ConnectorObjects property (Page 2199)

3.6.1.8 Properties

A

Actions Property

Description
Returns the Actions listing. Use the Actions property to configure an event-driven action.

Example:
In this example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle enlarges every time you click the button:

```vba
Sub CreateVBAActionToClickedEvent()
'VBA365
Dim objButton As HMIButton
Dim objCircle As HMICircle
Dim objEvent As HMIEvent
Dim objVBScript As HMIScriptInfo
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VB", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircle
    .Top = 100
    .Left = 100
    .BackColor = RGB(255, 0, 0)
End With
With objButton
    .Top = 10
    .Left = 10
    .Width = 120
    .Text = "Increase Radius"
End With
'Define event and assign sourcecode:
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
With objVBScript
    .SourceCode = "Dim myCircle" & vbCrLf & _
                "Set myCircle = HMIRuntime.ActiveScreen.ScreenItems(""Circle_VB"")" & _
                vbCrLf & "myCircle.Radius = myCircle.Radius + 5"
End With
End Sub
```
See also

Actions Object (Listing) (Page 1898)
AddAction Method (Page 1794)
Configuring Event-Driven Actions with VBA (Page 1699)

**ActionType property**

**Description**

Only used internally.

See also

VBA Reference: ActionDynamic (Page 1747)

**ActiveDocument Property**

**Description**

Returns an object of the "Document" type which represents the active picture in the Graphics Designer. If there is no open or active picture in the Graphics Designer, you receive an error message.

**Note**

The "ActiveDocument" property refers to the window that possesses the input focus. If other editors (e.g. CrossReference) access a picture, the input focus can change. To prevent this situation leading to errors, reference the picture unambiguously via the Documents listing.
Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles").

Sub CreateMenuItem()
'VBA366
Dim objMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
'
'Create new menu "Delete Objects"
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete Objects")
'
'Add two menuitems to the menu "Delete Objects"
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete Rectangles")
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete Circles")
End Sub

See also

Documents Object (Listing) (Page 1952)

ActiveLayer Property

Description

Defines or returns the active layer for the View object. The value range is from 0 to 31, where "0" represents the uppermost layer and "31" the lowest layer.

Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example a new View object is created and layer 1 is set to "Active":

Sub ActiveDocumentConfiguration()
'VBA367
Application.ActiveDocument.Views.Add
Application.ActiveDocument.Views(1).ActiveLayer = 2
End Sub

See also

View Object (Page 2102)
ActualPointLeft Property

Description

Defines or returns the X coordinate of the current corner point by reference to the picture origin (top left) for the objects "Polygon" and "Polyline". Each corner point is identified by an index which is derived from the number ("PointCount") of corner point available.

A change of the value can affect the properties "Width" (object width) and "Left" (x-coordinate of the object position).

Example:

The "PolygonCoordinatesOutput()" procedure outputs the coordinates of all the corner points in the first polyline in the current picture:

```vba
Sub PolygonCoordinatesOutput()
    'VBA368
    Dim objPolyline As HMIPolyLine
    Dim iPosX As Integer
    Dim iPosY As Integer
    Dim iCounter As Integer
    Dim strResult As String
    iCounter = 1
    Set objPolyline = ActiveDocument.HMIObjects.AddHMIObject("Polyline1", "HMIPolyLine")
    For iCounter = 1 To objPolyline.PointCount
        With objPolyline
            .index = iCounter
            iPosX = .ActualPointLeft
            iPosY = .ActualPointTop
        End With
        strResult = strResult & vbCrLf & "Corner " & iCounter & ": x=" & iPosX & ", y=" & iPosY
    Next iCounter
    MsgBox strResult
End Sub
```

See also

- PointCount Property (Page 2379)
- Index Property (Page 2276)
- ActualPointTop Property (Page 2110)
- PolyList Object (Page 2038)
- Polygon Object (Page 2035)
- Line Object (Page 2004)
ActualPointTop Property

Description

Defines or returns the Y coordinate of the current corner point by reference to the picture origin (top left) for the objects "Polygon" and "Polyline". Each corner point is identified by an index which is derived from the number ("PointCount") of corner point available.

A change of the value can affect the properties "Height" (object height) and "Top" (y-coordinate of the position).

Example:

The "Polygon()" procedure outputs the coordinates of all the corner points in the first polyline in the current picture:

```vba
Sub PolygonCoordinatesOutput()
    'VBA369
    Dim objPolyline As HMIPolyLine
    Dim iPosX As Integer
    Dim iPosY As Integer
    Dim iCounter As Integer
    Dim strResult As String
    iCounter = 1
    Set objPolyline = ActiveDocument.HMIObjects.AddHMIObject("Polyline1", "HMIPolyLine")
    For iCounter = 1 To objPolyline.PointCount
        With objPolyline
            .index = iCounter
            iPosX = .ActualPointLeft
            iPosY = .ActualPointTop
        End With
        strResult = strResult & vbCrLf & "Corner " & iCounter & ": x=" & iPosX & " y=" & iPosY
    Next iCounter
    MsgBox strResult
End Sub
```

See also

- PointCount Property (Page 2379)
- Index Property (Page 2276)
- ActualPointLeft Property (Page 2109)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- Line Object (Page 2004)
AdaptBorder Property

Description
TRUE if the field border is intended to adapt dynamically to the size of the text. BOOLEAN write-read access.

Note
Changing the contents of a field dynamically can cause pumping in the field. Performance is improved in Runtime by using "AdaptBorder = False".

Example:
The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the text size is dynamically adapted to the field size.

```vba
Sub IOFieldConfiguration()
'VBA372
Dim objIOField As HMIIOField
'Add new IO-Feld to active document:
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
 .AdaptBorder = True
End With
End Sub
```

See also
OptionGroup Object (Page 2025)
TextList Object (Page 2076)
StaticText Object (Page 2067)
IOField Object (Page 1992)
CheckBox Object (Page 1924)
Button Object (Page 1921)

AdaptPicture Property

Description
TRUE if the picture size is to be adapted to the picture window size. BOOLEAN write-read access.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

```vba
Sub PictureWindowConfig()
'VBA373
Dim objPicWindow As HMIPictureWindow
'
'Add new picturewindow into active document:
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIStory")
With objPicWindow
    .AdaptPicture = False
    .AdaptSize = False
    .Caption = True
    .CaptionText = "Picturewindow in runtime"
    .OffsetLeft = 5
    .OffsetTop = 10
    '
    'Replace the picturename "Test.PDL" with the name of
    'an existing document from your "GraCS"-Folder of your active project
    .PictureName = "Test.PDL"
    .ScrollBars = True
    .ServerPrefix = ""
    .TagPrefix = "Struct."
    .UpdateCycle = 5
    .Zoom = 100
End With
End Sub
```

See also

PictureWindow Object (Page 2029)

AdaptSize Property

Description

TRUE if the picture window size is to be adapted to the picture size. BOOLEAN write-read access.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

```vba
Sub PictureWindowConfig()
'VBA374
Dim objPicWindow As HMIPictureWindow

'Add new picturewindow into active document:
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10

  'Replace the picturename "Test.PDL" with the name of
  'an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ServerPrefix = ""
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub
```

See also

- PictureWindow Object (Page 2029)

AddIns property

Description

Only used internally.

See also

- Application Object (Page 1909)
AlarmGoneVisible property

Description

Defines whether an outgoing state is visible.

<table>
<thead>
<tr>
<th>Assigned Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>The outgoing state is visible.</td>
</tr>
<tr>
<td>False</td>
<td>The outgoing state is suppressed.</td>
</tr>
</tbody>
</table>

AlarmHigh Property

Description

Defines the top limit value at which an alarm should be triggered or returned. The type of the evaluation (in percent or absolute) is defined in the "TypeAlarmHigh" property. The "CheckAlarmHigh" property defines whether the monitoring function for the limit value is activated.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "50".

Sub BarGraphLimitConfiguration()
'VBA375
Dim objBarGraph As HMIBarGraph
'
'Add new BarGraph to active document:
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis to absolut
.TypeAlarmHigh = False
'Activate monitoring
.CheckAlarmHigh = True
'Activate barcolor to "yellow"
.ColorAlarmHigh = RGB(255, 255, 0)
'set upper limit to "50"
.AlarmHigh = 50
End With
End Sub
See also

TypeAlarmHigh Property (Page 2451)
ColorAlarmHigh Property (Page 2182)
CheckAlarmHigh Property (Page 2170)
BarGraph Object (Page 1914)

AlarmLow Property

Description

Defines the bottom limit value at which an alarm should be triggered or returned.
The type of the evaluation (in percent or absolute) is defined in the "TypeAlarmLow" property.
The "CheckAlarmLow" property defines whether the monitoring function for the limit value is activated.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "10".

Sub BarGraphLimitConfiguration()
  'VBA376
  Dim objBarGraph As HMIBarGraph
  'Add new BarGraph to active document:
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis to absolut
    .TypeAlarmLow = False
    'Activate monitoring
    .CheckAlarmLow = True
    'Set Barcolor to "yellow"
    .ColorAlarmLow = RGB(255, 255, 0)
    'set lower limit to "10"
    .AlarmLow = 10
  End With
End Sub

See also

TypeAlarmLow Property (Page 2452)
ColorAlarmLow Property (Page 2183)
Alignment Property

Description

Defines or returns the scale display (left/right or top/bottom) depending on the position of the BarGraph object. The Scaling property must be set to TRUE for the scale to be displayed.

<table>
<thead>
<tr>
<th>Display</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right or bottom</td>
<td>TRUE</td>
</tr>
<tr>
<td>Left or top</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Example:

The "BarGraphConfiguration()" procedure configures In this example the scale is to be located to the right of the bar:

Sub BarGraphConfiguration()
    'VBA377
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .Alignment = True
        .Scaling = True
    End With
End Sub

See also

Scaling Property (Page 2405)
Direction Property (Page 2213)
BarGraph Object (Page 1914)
AlignmentLeft Property

Description

Defines or returns the horizontal alignment of the text. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Horizontal Alignment</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>0</td>
</tr>
<tr>
<td>Centered</td>
<td>1</td>
</tr>
<tr>
<td>Right</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the text in the I/O field will be centered horizontally:

```vba
Sub IOFieldConfiguration()
    'VBA378
    Dim objIOField As HMIIOField
    Set objIOField = ActiveDocument.HMIOObjects.AddHMIObject("IOField1", "HMIIOField")
    With objIOField
        .AlignmentLeft = 1
    End With
End Sub
```

Related topics

See also

- AlignmentTop Property (Page 2118)
- TextList Object (Page 2076)
- StaticText Object (Page 2067)
- OptionGroup Object (Page 2025)
- GroupDisplay Object (Page 1980)
- IOField Object (Page 1992)
- CheckBox Object (Page 1924)
- Button Object (Page 1921)
AlignmentTop Property

Description

Defines or returns the vertical alignment of the text. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Horizontal Alignment</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>0</td>
</tr>
<tr>
<td>Centered</td>
<td>1</td>
</tr>
<tr>
<td>Down</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the text in the I/O field will be centered in the middle:

Sub IOFieldConfiguration()
'VBA379
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
  .AlignmentLeft = 1
  .AlignmentTop = 1
End With
End Sub

See also

AlignmentLeft Property (Page 2117)
TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
GroupDisplay Object (Page 1980)
IOField Object (Page 1992)
CheckBox Object (Page 1924)
Button Object (Page 1921)

AnalogResultInfos Property

Description

Returns the AnalogResultInfos listing. Use the AnalogResultInfos property to define value ranges and property values in the Dynamic dialog.
Example:

An example showing how to use the AnalogResultInfos property can be found in this documentation under the heading "AnalogResultInfos Object (Listing)".

See also

DynamicDialog Object (Page 1955)
AnalogResultInfos Object (Listing) (Page 1908)

AngleAlpha Property

Description

Defines or returns depth angle \( a \) for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example depth angles \( A \) and \( B \) will be assigned the values "15" and 45:

Sub HMI3DBarGraphConfiguration()
'VBA380
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjcts.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
'Depth-angle \( a \) = 15 degrees
.AngleAlpha = 15
'Depth-angle \( b \) = 45 degrees
.AngleBeta = 45
End With
End Sub
AngleBeta Property

Description

Defines or returns depth angle b for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example depth angles A and B will be assigned the values "15" and 45:

Sub HMI3DBarGraphConfiguration()
  'VBA381
  Dim obj3DBar As HMI3DBarGraph
  Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
  With obj3DBar
    'Depth-angle a = 15 degrees
    .AngleAlpha = 15
    'Depth-angle b = 45 degrees
    .AngleBeta = 45
  End With
End Sub

See also

AngleAlpha Property (Page 2119)
3DBarGraph Object (Page 1894)
**Application Property**

**Description**

Returns the Graphics Designer application when the application property is used without an object identifier. If the application property is used with object identifier, it returns an application object which displays the application with which the defined object was created. Read only access.

**Example:**

In this example an Excel object is created and the application name is output:

```vba
Sub CreateExcelApplication()
'VBA382
'Open Excel invisible
Dim objExcelApp As New Excel.Application
MsgBox objExcelApp
'Delete the reference to Excel and close it
Set objExcelApp = Nothing
End Sub
```

**See also**

Application Object (Page 1909)

**ApplicationDataPath Property**

**Description**

Returns the complete path of the active picture in the Graphics Designer. Read-only access.

**Example:**

The "ShowApplicationDataPath()" procedure outputs the path of the current picture:

```vba
Sub ShowApplicationDataPath()
'VBA383
MsgBox Application.ApplicationDataPath
End Sub
```
AssemblyInfo property

Description
Displays the information of the object registered in the Global Assembly Cache. The information is made up of "Assembly", "Version", "Culture" and "PublicKeyToken".

If the object is not registered in the Global Assembly Cache, the path of the object is only displayed in "Assembly".

Assignments Property

Description
A list which contains the assignments between the output values and the actual output texts to be output.

The assignments are dependent on the list type set. The list type is defined with the ListType property.

The number of entries depends on the total length of the string passed to the "Assignments" property. This string cannot be longer than 500,000 bytes. This may be checked prior to dropping access to the "Assignments" property by using the function LenB().

Example:

--

See also
ListType Property (Page 2312)
TextList Object (Page 2076)
AssumeOnExit Property

Description

TRUE, if the entered text is assumed after exiting the input field (by using the <TAB> key or mouse click, for example). BOOLEAN write-read access.

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the text that has been entered will be taken over as input on exit from the input field.

```vba
Sub IOFieldConfiguration()
 'VBA385
 Dim objIOField As HMIIOField
 Set objIOField = ActiveDocument.HMIOObjects.AddHMIObject("IOField1", "HMIIOField")
 With objIOField
   .AssumeOnExit = True
 End With
End Sub
```

See also

- TextList Object (Page 2076)
- IOField Object (Page 1992)

AssumeOnFull Property

Description

TRUE, when the content of the input field is full (specified number of characters have been entered) and should be exited automatically and the input accepted. BOOLEAN write-read access.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the text that has been entered will be taken over as input on exit from the input field.

Sub IOFieldConfiguration()
'VBA386
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIOObjects.AddHMIOObject("IOField1", "HMIIOField")
With objIOField
  .AssumeOnFull = True
End With
End Sub

See also

OutputFormat Property (Page 2357)
DataFormat Property (Page 2210)
IOField Object (Page 1992)

AutomationName Property

Description

Depending on the source and destination object types for the direct connection, either defines or returns the name of a property.

The two tables show you when you must use the AutomationName property. A "--" means that the property is assigned an empty string ("") by default when the DirectConnection object is created.

Source object type (SourceLink Property)

<table>
<thead>
<tr>
<th>Type Property</th>
<th>AutomationName Property</th>
<th>ObjectName Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmiSourceTypeConstant</td>
<td>--</td>
<td>Name of the constant (e.g. the picture name)</td>
</tr>
<tr>
<td>hmiSourceTypeProperty</td>
<td>Property of the source object (e.g. &quot;Top&quot;)</td>
<td>Name of the source object (e.g. &quot;Rectangle_A&quot;)</td>
</tr>
<tr>
<td>hmiSourceTypePropertyOfThisObject</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>hmiSourceTypeVariableDirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiSourceTypeVariableIndirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
</tbody>
</table>
### Destination object type (DestinationLink Property)

<table>
<thead>
<tr>
<th>Type Property</th>
<th>AutomationName Property</th>
<th>ObjectName Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmiDestTypeProperty</td>
<td>Property of the destination object (e.g. &quot;Left&quot;)</td>
<td>Name of the destination object (e.g. &quot;Rectangle_A&quot;)</td>
</tr>
<tr>
<td>hmiDestTypePropertyOfThisObject</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>hmiDestTypePropertyOfActualWindow</td>
<td>Property of the destination object (e.g. &quot;Left&quot;)</td>
<td>--</td>
</tr>
<tr>
<td>hmiDestTypeVariableDirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeVariableIndirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeDirectMessage</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeIndirectMessage</td>
<td>--</td>
<td>Tag name</td>
</tr>
</tbody>
</table>
Example:

In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
'VBA387
Dim objButton As HMIButton
Dim objRectangleA As HMIRectangle
Dim objRectangleB As HMIRectangle
Dim objEvent As HMIEvent
Dim objDynConnection As HMIDirectConnection

'Add objects to active document:
Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")

'to position and configure objects:
With objRectangleA
 .Top = 100
 .Left = 100
End With
With objRectangleB
 .Top = 250
 .Left = 400
 .BackColor = RGB(255, 0, 0)
End With
With objButton
 .Top = 10
 .Left = 10
 .Text = "SetPosition"
End With

'Directconnection is initiate by mouseclick:
Set objDynConnection =
objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
With objDynConnection
 'Sourceobject: Top-Property of Rectangle_A
 .SourceLink.Type = hmiSourceTypeProperty
 .SourceLink.ObjectName = "Rectangle_A"
 .SourceLink.AutomationName = "Top"
 ,
 'Targetobject: Left-Property of Rectangle_B
 .DestinationLink.Type = hmiDestTypeProperty
 .DestinationLink.ObjectName = "Rectangle_B"
 .DestinationLink.AutomationName = "Left"
End With
End Sub
AvailableDataLanguages Property

Description

Returns a listing of the available project languages.

Example:

The "AusgabetDataLanguages()" procedure outputs all the existing project languages together with their language identifiers (as a decimal value):

```vba
Sub OutputDataLanguages()
'VBA388
Dim colDataLang As HMIDataLanguages
Dim objDataLang As HMIDataLanguage
Dim strLangList As String
Dim iCounter As Integer
'
'Save collection of datalanguages
'into variable "colDataLang"
Set colDataLang = Application.AvailableDataLanguages
iCounter = 1
'
'Get every languagename and the assigned ID
For Each objDataLang In colDataLang
With objDataLang
If 0 = iCounter Mod 3 Or 1 = iCounter Then
    strLangList = strLangList & vbCrLf & .LanguageID & " " & .LanguageName
Else
    strLangList = strLangList & " / " & .LanguageID & " " & .LanguageName
End If
End With
iCounter = iCounter + 1
Next objDataLang
MsgBox strLangList
End Sub
```
Average Property

Description

TRUE, if the mean value is calculated based on the last 10 values. A value change is conditional for calculation of a new mean value. The mean value is reset when you change a picture. If only one value is available when you change the picture, the following mean value is calculated: \( \frac{5+0+0+0+0+0+0+0+0+0}{10} = 0,5 \).

BOOLEAN write-read access.

Example

The "BarGraphConfiguration()" procedure configures In this example, value averaging will be activated:

```vba
Sub BarGraphConfiguration()
    'VBA389
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .Average = True
    End With
End Sub
```

See also

BarGraph Object (Page 1914)
Axe Property

Description

Defines or returns the axis for displaying the measured value. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the Y axis for displaying the measured value will be defined:

```vba
Sub HMI3DBarGraphConfiguration()
  'VBA390
  Dim obj3DBar As HMI3BarGraph
  Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
  With obj3DBar
    .Axe = 1
  End With
End Sub
```

See also

3DBarGraph Object (Page 1894)
AxisSection Property

Description
Defines or returns the distance between two long axis sections. The information on the distance is given in scale units and is dependent on the minimum and maximum values configured.

![BarGraph Object (Minimum/Maximum Value: -5/5; AxisSection = 2)](image)

Example
The "BarGraphConfiguration()" procedure accesses the properties of the BarGraph object. In this example the axis section will be set to "2".

```vba
Sub BarGraphConfiguration()
    'VBA391
    Dim objBar As HMIBarGraph
    Set objBar = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBar
        .AxisSection = 2
    End With
End Sub
```

See also
BarGraph Object (Page 1914)
BackBorderWidth Property

Description
Defines or returns the width of the 3D border in pixels. The value for the width is dependent on the size of the object.

Slider
Defines or returns the width of the border in pixels. BackBorderWidth = 0 prevents the border being displayed on the Slider object.

Example:
The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the width of the 3D border will be set to "2".

```vba
Sub ButtonConfiguration()
  'VBA392
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjec.ts.AddHMIObject("Button1", "HMIButton")
  With objButton
    .BackBorderWidth = 2
  End With
End Sub
```

See also
- Slider object (Page 2062)
- RoundButton Object (Page 2052)
- GroupDisplay Object (Page 1980)
- Button Object (Page 1921)

BackColor Property

Description
Defines or returns the background color for the object. LONG read-write access.
The background color is not displayed if "transparent" is defined as the fill pattern.
Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the background color will be set to "Yellow".

Sub RectangleConfiguration()
'VBA393
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
 .BackColor = RGB(255, 255, 0)
End With
End Sub

See also

EllipseSegment Object (Page 1963)
StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
GroupDisplay Object (Page 1980)
GraphicObject Object (Page 1975)
IOField Object (Page 1992)
Ellipse Object (Page 1957)
Document Object (Page 1949)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BackColor2 Property

Description

Defines or returns the bar color for the display of the current value. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "BarGraphConfiguration()" procedure configures In this example the bar color for displaying the current value will be set to "Yellow":

```vba
Sub BarGraphConfiguration()
  'VBA394
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    .BackColor2 = RGB(255, 255, 0)
  End With
End Sub
```

See also

BarGraph Object (Page 1914)

BackColor3 Property

Description

Defines or returns the color of the bar background. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

**Example:**

The "BarGraphConfiguration()" procedure configures In this example the color of the bar background will be set to "Blue":

```vba
Sub BarGraphConfiguration()
'VBA395
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  .BackColor3 = RGB(0, 0, 255)
End With
End Sub
```

**See also**

BarGraph Object (Page 1914)

**BackColor_Alarm..._Warning property**

**Description**

Defines the color used for the background of one of the following states or message types:

- Alarm
- Warning
- Tolerance
- AS Process Control Error
- AS Control System Fault
- Operator request
- OK
- Simulation

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
BackColorBottom Property

Description
Defines or returns the color for the bottom/right part of the slider. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the color of the bottom part of the slider will be set to "Blue":

Sub SliderConfiguration()
    'VBA396
    Dim objSlider As HMISlider
    Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
    With objSlider
        .BackColorBottom = RGB(0, 0, 255)
    End With
End Sub

See also
Slider object (Page 2062)

BackColorTop Property

Description
Defines or returns the color for the top/left part of the slider. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the color of the top part of the slider will be set to "Yellow":

```vba
Sub SliderConfiguration()
'VBA397
Dim objSlider As HMISlider
Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
With objSlider
  .BackColorTop = RGB(255, 255, 0)
End With
End Sub
```

See also

Slider object (Page 2062)

**BackFillColor property**

**Description**

Defines the color with which the background is filled at an advanced analog display.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

**BackFillColor_OK property**

**Description**

Defines the color with which the background is filled at the state "OK".

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
BackFillColor_Simulation property

Description
Defines the color with which the background is filled at the "Simulation" state.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

BackFillStyle property

Description
Defines the pattern with which the background is filled at an advanced analog display.

There is a choice of 50 fill patterns. The fill pattern 0 "Solid" fills the object with the set background color; the fill pattern 1 "Transparent" defines that neither a background nor a fill pattern is displayed.

BackFillStyle_OK property

Description
Defines the pattern with which the background is displayed at the state "OK".

There is a choice of 50 fill patterns. The fill pattern 0 "Solid" fills the object with the set background color; the fill pattern 1 "Transparent" defines that neither a background nor a fill pattern is displayed.

BackFillStyle_Simulation property

Description
Defines the pattern with which the background is displayed at the "Simulation" state.

There is a choice of 50 fill patterns. The fill pattern 0 "Solid" fills the object with the set background color; the fill pattern 1 "Transparent" defines that neither a background nor a fill pattern is displayed.
BackFlashColorOff Property

Description

Defines or returns the color of the object background for the flash status "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the color when the flash status is "Off" will be set to "Yellow":

```
Sub RectangleConfiguration()
    'VBA398
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjct.AddHMIObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .BackFlashColorOff = RGB(255, 255, 0)
    End With
End Sub
```

See also

BarGraph Object (Page 1914)
StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
GraphicObject Object (Page 1975)
IOField Object (Page 1992)
EllipseSegment Object (Page 1963)
BackFlashColorOn Property

Description
Defines or returns the color of the object background for the flash status "On". LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the color when the flash status is "On" will be set to "Blue":

```vba
Sub RectangleConfiguration()
  'VBA399
  Dim objRectangle As HMIRectangle
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
  With objRectangle
    .BackFlashColorOn = RGB(0, 0, 255)
  End With
End Sub
```

See also
RoundButton Object (Page 2052)
StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
Background Property

Description

TRUE, when the background of the 3D-bar graph object should be visible. BOOLEAN write-read access.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the background will be set to "Transparent":

```vba
Sub HMI3DBarGraphConfiguration()
  'VBA400
  Dim obj3DBar As HMI3DBarGraph
  Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
  With obj3DBar
    .Background = False
  End With
End Sub
```

See also

3DBarGraph Object (Page 1894)
BackPictureAlignment property

Description
As the "Display type" attribute, defines the position and scaling for the background image of the process picture.

- **normal**: The background picture is centered in the original size. When opening the picture in runtime, it remains in the location.
- **Stretched (window)**: The background picture is scaled to the runtime window and process picture of the larger of the two windows. In runtime, it is scaled to the size of the runtime window and is scaled when you resize the picture.
- **Tiled**: Graphics Designer and process picture are exhibited with the picture in its original size.
- **Stretched (picture)**: The background picture is scaled to the configured size of the process picture. When opening the picture in runtime, it retains its size.

BackPictureName property

Description
Defines or returns the path and name of the file used as the background image in the process picture.

Files of format EMF, WMF, DB, BMP, GIF, JPG, JPEG and ICO are suitable.

If no path is specified, the file is searched for in the subdirectory \GraCS. If you specify a different path, a copy is created in the \GraCS directory.

Path specifications
The following path specification formats are possible:

- **Absolute**: z.B. "C:\Siemens\WinCC\Icons\myIcon.ICO.
- **Relative**: The starting folder for relative path specification is the "GraCS" folder of the current project.
- **<global>**: Refers to the installation path for WinCC. The path specification "<global>\Icons \myIcon" is the same as the path specification under "Absolute".
- **<project>**: Refers to the current project directory.

BarDepth Property

Description
Defines or returns the depth of the bar in pixels.
Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the bar depth will be set to "40":

Sub HMI3DBarGraphConfiguration()
'VBA401
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
  .BarDepth = 40
End With
End Sub

See also
3DBarGraph Object (Page 1894)

BarHeight Property

Description
Defines or returns the height of the bar in pixels.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the bar height will be set to "60":

Sub HMI3DBarGraphConfiguration()
'VBA402
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
  .BarHeight = 60
End With
End Sub

See also
3DBarGraph Object (Page 1894)
BarWidth Property

Description

Defines or returns the width of the bar in pixels.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the bar width will be set to "80":

```vba
Sub HMI3DBarGraphConfiguration()
    'VBA403
    Dim obj3DBar As HMI3DBarGraph
    Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
    With obj3DBar
        .BarWidth = 80
    End With
End Sub
```

See also

3DBarGraph Object (Page 1894)

BasePicReferenced Property

Description

TRUE, when the picture assigned in the object status display should be saved. Otherwise, only the associated object reference is saved. BOOLEAN write-read access.

Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the picture assigned in the Status Display object is to be saved.

```vba
Sub StatusDisplayConfiguration()
    'VBA404
    Dim objStatDisp As HMIStatusDisplay
    Set objStatDisp = ActiveDocument.HMIObjects.AddHMIObject("Statusdisplay1", "HMIStatusDisplay")
    With objStatDisp
        .BasePicReferenced = True
    End With
End Sub
```
See also

StatusDisplay Object (Page 2070)

BasePicTransColor Property

Description

Defines or returns which color of the assigned bitmap object (.bmp, .dib) should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "BasePicUseTransColor" property is "True".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the color "Yellow" will be set to "Transparent".

Sub StatusDisplayConfiguration()
'VBA405
Dim objStatDisp As HMIStatusDisplay
Set objStatDisp = ActiveDocument.HMIObjects.AddHMIObject("Statusdisplay1", "HMIStatusDisplay")
With objStatDisp
 .BasePicTransColor = RGB(255, 255, 0)
 .BasePicUseTransColor = True
End With
End Sub

See also

BasePicUseTransColor Property (Page 2145)
StatusDisplay Object (Page 2070)

BasePicture Property

Description

Defines or returns the basic picture for the Status Display object.
The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated. In this context, the "BasePicReferenced" property defines whether the basic picture should be saved together with the object status display or referenced.

**Example:**

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the picture "Testpicture.BMP" will be used as the basic picture:

```vba
Sub StatusDisplayConfiguration()
  'VBA406
  Dim objStatDisp As HMIStatusDisplay
  Set objStatDisp = ActiveDocument.HMIObjects.AddHMIObject("Statusdisplay1", "HMIStatusDisplay")
  With objStatDisp
    'To use this example copy a Bitmap-Graphic
to the "GraCS"-Folder of the actual project.
    'Replace the picturename "Testpicture.BMP" with the name of
    'the picture you copied
    .BasePicture = "Testpicture.BMP"
  End With
  End Sub
```

**See also**

- BasePicReferenced Property (Page 2143)
- StatusDisplay Object (Page 2070)

**BasePicture property**

**Description**

Specifies which picture is to be displayed for the currently selected status. Pictures with the following formats can be inserted: EMF, WMF, BMP, GIF, JPG.

If no picture that you want to display is defined for a status, the symbol for the status display is shown as a placeholder.

**BasePicUseTransColor Property**

**Description**

TRUE, when the configured color ("BasePicTransColor" property) of the bitmap objects should be set to "transparent". BOOLEAN write-read access.
Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the color "Yellow" will be set to "Transparent":

Sub StatusDisplayConfiguration()
'VBA407
Dim objStatDisp As HMIStatusDisplay
Set objStatDisp = ActiveDocument.HMIObjects.AddHMIObject("Statusdisplay1", "HMIStatusDisplay")
With objStatDisp
  .BasePicTransColor = RGB(255, 255, 0)
  .BasePicUseTransColor = True
End With
End Sub

See also

BasePicTransColor Property (Page 2144)
StatusDisplay Object (Page 2070)

BaseX Property

Description

Defines or returns for the 3DBarGraph object the horizontal distance in pixels between the right-hand border of the bar and the left-hand border of the object field.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the horizontal distance will be set to "80".

Sub HMI3DBarGraphConfiguration()
'VBA408
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
  .BaseX = 80
End With
End Sub

See also

3DBarGraph Object (Page 1894)
BaseY Property

Description
Defines or returns for the 3DBarGraph object the vertical distance in pixels between the lower border of the bar and the upper border of the object field.

Example:
The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the vertical distance will be set to "100".

```vba
Sub HMI3DBarGraphConfiguration()
'VBA409
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjets.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
 .BaseY = 100
End With
End Sub
```

See also
3DBarGraph Object (Page 1894)

BinaryResultInfo Property

Description
Returns the BinaryResultInfo object.

Example:
An example showing how to use the BinaryResultInfo property can be found in this documentation under the heading "BinaryResultInfo Object".

See also
BinaryResultInfo Object (Page 1918)
**BitNotSetValue Property**

**Description**

Defines or returns the value for the dynamic property if the specified bit of a configured tag is not set.

To define which bit must be set in order to trigger a change of value, use the BitNumber property.

**Example:**

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned, the bit to be set will be defined and the associated property values will be assigned to the "set"/"not set" states:

```vba
Sub AddDynamicDialogToCircleRadiusTypeBit()
    'VBA410
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_B", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeBit
        .Trigger.VariableTriggers(1).CycleType = hmiVariableCycleType_5s
        .BitResultInfo.BitNumber = 1
        .BitResultInfo.BitSetValue = 40
        .BitResultInfo.BitNotSetValue = 80
    End With
End Sub
```

**See also**

- BitNumber Property (Page 2148)
- BitResultInfo Object (Page 1919)

**BitNumber Property**

**Description**

Defines or returns the bit whose status must change in order to trigger a change of value. The tag used must be of the type BYTE, WORD or DWORD.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic
dialog, a tag name will be assigned, the bit to be set will be defined and the associated property
values will be assigned to the "set"/"not set" states:

Sub AddDynamicDialogToCircleRadiusTypeBit()
    'VBA411
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_B", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeBit
        .BitResultInfo.BitNumber = 1
        .BitResultInfo.BitSetValue = 40
        .BitResultInfo.BitNotSetValue = 80
    End With
End Sub

See also

BitResultInfo Object (Page 1919)
VBA Reference (Page 1745)

BitPosition0..3 property

Description

Specifies the bit position of the selected tag for which the respective bit (0 to 3) of the status
value is used. The content is only evaluated when a tag is selected for the respective property
"BitSelect0..3". The tags are specified using "Process" and "Process1..3".

You can enter a value from "0" to "31". Each value can only be assigned once.

BitResultInfo Property

Description

Returns the BitResultInfo object.

Example:

An example showing how to use the BitResultInfo property can be found in this documentation
under the heading "BitResultInfo Object".
See also

BitResultInfo Object (Page 1919)

BitSelect0..3 property

Description

Specifies the status tag for which the respective (first to fourth) bit of the status value is specified. The tags are specified using the properties "Process" and "Process1..3".

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The respective bit of the status value is not evaluated. No status tag is used.</td>
</tr>
<tr>
<td>1</td>
<td>Status tag &quot;Process&quot; is used for the respective bit of the status value.</td>
</tr>
<tr>
<td>2</td>
<td>Status tag &quot;Process1&quot; is used for the respective bit of the status value.</td>
</tr>
<tr>
<td>3</td>
<td>Status tag &quot;Process2&quot; is used for the respective bit of the status value.</td>
</tr>
<tr>
<td>4</td>
<td>Status tag &quot;Process3&quot; is used for the respective bit of the status value.</td>
</tr>
</tbody>
</table>

BitSetValue Property

Description

Defines or returns the value for the dynamic property if the specified bit of a configured tag is set.

To define which bit must be set in order to trigger a change of value, use the BitNumber property.

Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned, the bit to be set will be defined and the associated property values will be assigned to the "set"/"not set" states:

```vba
Sub AddDynamicDialogToCircleRadiusTypeBit()
'VBA412
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_B", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
With objDynDialog
   .ResultType = hmiResultTypeBit
   .BitResultInfo.BitNumber = 1
   .BitResultInfo.BitSetValue = 40
   .BitResultInfo.BitNotSetValue = 80
End With
End Sub
```
See also

- BitNumber Property (Page 2148)
- BitResultInfo Object (Page 1919)

**Bold Property**

**Description**

TRUE if the font attribute "Bold" is set for the language-dependent text in the object. BOOLEAN write-read access.

**Example:**

```vba
Sub ExampleForLanguageFonts()
'VBA413
Dim colLangFonts As HMILanguageFonts
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
objButton.Text = "Displaytext"
Set colLangFonts = objButton.LDFonts
'Set french fontproperties:
With colLangFonts.ItemByLCID(1036)
  .Family = "Courier New"
  .Bold = True
  .Italic = False
  .Underlined = True
  .Size = 12
End With

'Set english fontproperties:
With colLangFonts.ItemByLCID(1033)
  .Family = "Times New Roman"
  .Bold = False
  .Italic = True
  .Underlined = False
  .Size = 14
End With
End Sub
```

**Note**

For this example to work, you must already have configured in the languages concerned.
BorderBackColor Property

Description
Defines or returns the background color of the line for the object. LONG write-read access. The background color is only visible if the BorderStyle property is set >0.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the background color for the line will be set to "Yellow":

Sub RectangleConfiguration()
'VBA415
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.BorderBackColor = RGB(255, 255, 0)
End With
End Sub

See also
PieSegment Object (Page 2032)
BorderStyle Property (Page 2160)
BorderColor Property

Description

Defines or returns the line color for the object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the line color will be set to "Blue".

```vba
Sub RectangleConfiguration()
'VBA416
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
    .BorderColor = RGB(0, 0, 255)
End With
End Sub
```

See also

GraphicObject Object (Page 1975)
TextList Object (Page 2076)
StatusDisplay Object (Page 2070)
StaticText Object (Page 2067)
Slider object (Page 2062)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
PolyLine Object (Page 2038)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
Line Object (Page 2004)
IOField Object (Page 1992)
EllipseArc Object (Page 1960)
EllipseSegment Object (Page 1963)
Ellipse Object (Page 1957)
CircularArc Object (Page 1931)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BarGraph Object (Page 1914)
BorderColorBottom Property

Description
Defines or returns the color for the bottom right-hand part of the 3D-border. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the 3D-border color will be defined:

```
Sub ButtonConfiguration()
'VBA417
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
    .BorderColorBottom = RGB(255, 0, 0)
    .BorderColorTop = RGB(0, 0, 255)
End With
End Sub
```

See also
RoundButton Object (Page 2052)
Button Object (Page 1921)

BorderColorTop Property

Description
Defines or returns the color for the top left-hand part of the 3D-border. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the 3D-border color will be defined:

```
Sub ButtonConfiguration()
    'VBA418
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
    With objButton
        .BorderColorBottom = RGB(255, 0, 0)
        .BorderColorTop = RGB(0, 0, 255)
    End With
End Sub
```

See also

RoundButton Object (Page 2052)
Button Object (Page 1921)

BorderEndStyle Property

Description

Defines or returns the line end style of the object. LONG read-write access.

Determination of Line End Style

Determine the line end type with the aid of a five-character hexadecimal value which you then convert into its equivalent decimal value.
To determine the line ends for the object, go to the "Line End Style" window and proceed as follows:

- **Left column**: Configures the start of the line. Value range (from the top down) 0 to 6. The start of the line corresponds to the first character in the hexadecimal value. In the configuration shown, the value of the first character is "3".
- **Right column**: Configures the end of the line. Value range (from the top down) 0 to 6. The line end corresponds to the fifth character in the hexadecimal value. In the configuration shown, the value of the fifth character is "6".

This gives a hexadecimal value of "60003". This corresponds to a decimal value of "393219", which you then assign to the BorderEndStyle property.

**Example:**

The "LineConfiguration()" procedure accesses the properties of the line. In this example the type of line end will be set to the configuration illustrated above:

```vba
Sub LineConfiguration()
    'VBA419
    Dim objLine As HMILine
    Set objLine = ActiveDocument.HMIObjects.AddHMIObject("Line1", "HMILine")
    With objLine
        .BorderEndStyle = 393219
    End With
End Sub
```

**See also**

- PolyLine Object (Page 2038)
- Line Object (Page 2004)

**BorderFlashColorOff Property**

**Description**

Defines or returns the color of the object lines for the flashing status "Off". LONG write-read access.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the color when the flash status is "Off" will be set to "Black":

```vba
Sub RectangleConfiguration()
'VBA420
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
 .BorderFlashColorOff = RGB(0, 0, 0)
End With
End Sub
```

See also

- RoundButton Object (Page 2052)
- StatusDisplay Object (Page 2070)
- StaticText Object (Page 2067)
- Slider object (Page 2062)
- TextList Object (Page 2076)
- RoundRectangle Object (Page 2055)
- Rectangle Object (Page 2049)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- PieSegment Object (Page 2032)
- OptionGroup Object (Page 2025)
- Line Object (Page 2004)
- GraphicObject Object (Page 1975)
- IOField Object (Page 1992)
- EllipseSegment Object (Page 1963)
- EllipseArc Object (Page 1960)
- Ellipse Object (Page 1957)
- CircularArc Object (Page 1931)
- Circle Object (Page 1928)
- CheckBox Object (Page 1924)
- Button Object (Page 1921)
BorderFlashColorOn Property

Description

Defines or returns the color of the object lines for the flashing status "On". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the color when the flash status is "On" will be set to "Red":

```vba
Sub RectangleConfiguration()
  'VBA421
  Dim objRectangle As HMIRectangle
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
  With objRectangle
    .BorderFlashColorOn = RGB(255, 0, 0)
  End With
End Sub
```

See also

- StaticText Object (Page 2067)
- StatusDisplay Object (Page 2070)
- Slider object (Page 2062)
- TextList Object (Page 2076)
- RoundRectangle Object (Page 2055)
- RoundButton Object (Page 2052)
- Rectangle Object (Page 2049)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- PieSegment Object (Page 2032)
- OptionGroup Object (Page 2025)
- Line Object (Page 2004)
- GraphicObject Object (Page 1975)
BorderStyle Property

Description

Defines or returns the line style for the object. Value range from 0 to 4:

<table>
<thead>
<tr>
<th>Line style</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>- - - - - -</td>
<td>0</td>
</tr>
<tr>
<td>- - - -</td>
<td>1</td>
</tr>
<tr>
<td>- - -</td>
<td>2</td>
</tr>
<tr>
<td>- -</td>
<td>3</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the line style will be set to "1":

```vba
Sub RectangleConfiguration()
    'VBA422
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIOObjects.AddHMIOObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .BorderStyle = 1
    End With
End Sub
```

See also

IOField Object (Page 1992)
StatusDisplay Object (Page 2070)
StaticText Object (Page 2067)
BorderWidth Property

Description

Defines or returns the line weight (in pixels) for the object.

Example:

in the following example the line weight of a newly added circle will be set to "2".

Sub CircleConfiguration()
    'VBA423
    Dim objCircle As IHMICircle
    Set objCircle = ActiveDocument.HMIObjec()AddHMIObject("Circle1", "HMICircle")
    With objCircle
        .BorderWeight = 2
    End With
End Sub
See also

IOField Object (Page 1992)
StatusDisplay Object (Page 2070)
StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
PolyLine Object (Page 2038)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
Line Object (Page 2004)
GraphicObject Object (Page 1975)
EllipseSegment Object (Page 1963)
EllipseArc Object (Page 1960)
Ellipse Object (Page 1957)
CircularArc Object (Page 1931)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)

**BottomConnectedObjectName Property**

**Description**

Returns the name of the starting object to which the connector is Read only access.

**Example:**

An example showing how to use the BottomConnectedObjectName property can be found in this documentation under the heading "ObjConnection Object".

**See also**

ObjConnection object (Page 2019)
**BottomConnectedConnectionPointIndex Property**

**Description**

Returns the connection point on the object to which the connector is connected.

<table>
<thead>
<tr>
<th>Connection Point</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>0</td>
</tr>
<tr>
<td>Right</td>
<td>1</td>
</tr>
<tr>
<td>Down</td>
<td>2</td>
</tr>
<tr>
<td>Left</td>
<td>3</td>
</tr>
</tbody>
</table>

**Example:**

An example showing how to use the BottomConnectedObjectName property can be found in this documentation under the heading "ObjConnection Object".

**See also**

ObjConnection object (Page 2019)

**BoxAlignment Property**

**Description**

TRUE, when the fields are arranged aligned to the right. BOOLEAN write-read access.
Example:

The "CreateOptionGroup()" procedure creates the OptionGroup object with four option buttons. Each option button is assigned the default name "myCustomText<Nummer>".

Sub CreateOptionGroup()
' VBA424
Dim objRadioBox As HMIOptionGroup
Dim iCounter As Integer
Set objRadioBox = ActiveDocument.HMIObjects.AddHMIObject("RadioBox_1", "HMIOptionGroup")
iCounter = 1
With objRadioBox
 .Height = 100
 .Width = 180
 .BoxCount = 4
 .BoxAlignment = False
 For iCounter = 1 To .BoxCount
  .index = iCounter
  .Text = "CustomText" & .index
 Next iCounter
End With
End Sub

See also

BoxCount Property (Page 2164)
OptionGroup Object (Page 2025)
CheckBox Object (Page 1924)

BoxCount Property

Description

Defines or returns the number of fields. Value range from 1 to 32.
Example:

The "CreateOptionGroup()" procedure creates the OptionGroup object with four option buttons. Each option button is assigned the default name "myCustomText<Nummer>".

```vba
Sub CreateOptionGroup()
  'VBA425
  Dim objRadioBox As HMIOptionGroup
  Dim iCounter As Integer
  Set objRadioBox = ActiveDocument.HMIObjects.AddHMIObject("RadioBox_1", "HMIOptionGroup")
  iCounter = 1
  With objRadioBox
    .Height = 100
    .Width = 180
    .BoxCount = 4
    .BoxAlignment = True
    For iCounter = 1 To .BoxCount
      .index = iCounter
      .Text = "CustomText" & .index
    Next iCounter
  End With
End Sub
```

See also

BoxAlignment Property (Page 2163)
OptionGroup Object (Page 2025)
CheckBox Object (Page 1924)

BoxType Property

Description

 Defines or returns the field type. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Field type</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition</td>
<td>0</td>
</tr>
<tr>
<td>Input</td>
<td>1</td>
</tr>
<tr>
<td>I/O field</td>
<td>2</td>
</tr>
</tbody>
</table>
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the field type is configured as "Input":

Sub IOFieldConfiguration()
'VBA426
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.BoxType = 1
End With
End Sub

See also

IOField Object (Page 1992)

Button1..8MessageClasses

Description

Defines one or more message events in the group display for representing the respective command button. This is done by entering the numbers of the bits in the collective value.

If you want to assign several message events, delimit the numbers with a comma. The order of assignment defines the priority. If there is more than one selected event for one button, the event that has been entered first is displayed.

The same event can be visualized simultaneously in several buttons.

Button1..8Width property

Description

Defines or returns for the "Group Display" object the width of the respective button in pixels. When the "SameSize" property is set to "TRUE", all buttons are set to the same width.
Example

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the width of button "1" is set to "50":

```vba
Sub GroupDisplayConfiguration()
  'VBA427
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .Button1Width = 50
  End With
End Sub
```

See also

- SameSize Property (Page 2403)
- GroupDisplay Object (Page 1980)

ButtonColor Property

Description

Defines or returns the color of the slider for the Slider object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the color of the slider will be set to "Yellow":

```vba
Sub SliderConfiguration()
  'VBA431
  Dim objSlider As HMISlider
  Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
  With objSlider
    .ButtonColor = RGB(255, 255, 0)
  End With
End Sub
```
See also

Slider object (Page 2062)

C

Caption Property

Description

TRUE, when the application or picture window has a title bar in Runtime. BOOLEAN write-read access.

The Caption property must be set to "True" if the intention is that the application window or picture window shall have Maximize and Close buttons.

Example:

The "ApplicationWindowConfig" procedure accesses the properties of the application window. In this example the application window will

Sub ApplicationWindowConfig()
    'VBA432
    Dim objAppWindow As HMIApplicationWindow
    Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow", "HMIApplicationWindow")
    With objAppWindow
        .Caption = True
        .CloseButton = False
        .Height = 200
        .Left = 10
        .MaximizeButton = True
        .Moveable = False
        .OnTop = True
        .Sizeable = True
        .Top = 20
        .Visible = True
        .Width = 250
        .WindowBorder = True
    End With
End Sub

See also

PictureWindow Object (Page 2029)
ApplicationWindow Object (Page 1911)
CaptionText Property

Description
Defines or returns the window title that will be displayed for the PictureWindow object in Runtime.
The Caption property must be set to TRUE."

Example:
The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will

```
Sub PictureWindowConfig()
  'VBA433
  Dim objPicWindow As HMIPictureWindow
  Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
  With objPicWindow
    .AdaptPicture = False
    .AdaptSize = False
    .Caption = True
    .CaptionText = "Picturewindow in runtime"
    .OffsetLeft = 5
    .OffsetTop = 10
    'Replace the picturename "Test.PDL" with the name of
    'an existing document from your "GraCS"-Folder of your active project
    .PictureName = "Test.PDL"
    .ScrollBars = True
    .ServerPrefix = ""
    .TagPrefix = "Struct."
    .UpdateCycle = 5
    .Zoom = 100
  End With
End Sub
```

See also
PictureWindow Object (Page 2029)

CBackColorOff..ColorOn property

Description
Specifies for the selected message type and the state "Came In" which color the background of the value to be displayed assumes for flashing status "Off" (CBackColorOff) or "On" (CBackColorOn).
Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

CBackFlash property

Description

Specifies for the selected message type and status "Came In" whether the background of the value to be displayed flashes when a message is received.

CheckAlarmHigh Property

Description

TRUE if the "Alarm High" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties AlarmHigh, ColorAlarmHigh and TypeAlarmHigh.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "50".

Sub BarGraphLimitConfiguration()
'VBA434
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis to absolute
  .TypeAlarmHigh = False
  'Activate monitoring
  .CheckAlarmHigh = True
  'Set barcolor to "yellow"
  .ColorAlarmHigh = RGB(255, 255, 0)
  'Set upper limit to "50"
  .AlarmHigh = 50
End With
End Sub
CheckAlarmLow Property

Description
TRUE if the "Alarm Low" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties AlarmLow, ColorAlarmLow and TypeAlarmLow.

Example:
The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "10".

Sub BarGraphLimitConfiguration()
    'VBA435
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .TypeAlarmLow = False
        .Activate monitoring
        .CheckAlarmLow = True
        .Set barcolor to "yellow"
        .ColorAlarmLow = RGB(255, 255, 0)
        .Set lower limit to "10"
        .AlarmLow = 10
    End With
End Sub

See also
ColorAlarmLow Property (Page 2183)
TypeAlarmLow Property (Page 2452)
AlarmLow Property (Page 2115)
BarGraph Object (Page 1914)
**Checked Property**

**Description**

TRUE if a check mark is to appear in front of the user-defined menu entry. BOOLEAN write-read access.

**Example:**

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"): The first menu entry is also marked with a tick:

```vba
Sub CreateMenuItem()
    'VBA436
    Dim objMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem

    'Add new menu "Delete objects" to menubar:
    Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")

    'Add two menuitems to the new menu
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete Rectangles")
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete Circles")
    With objMenu.MenuItems
        .Item("DeleteAllRectangles").Checked = True
    End With
End Sub
```

**See also**

- MenuItems Property (Page 2337)
- Configuring Menus and Toolbars (Page 1626)

**CheckLimitHigh4 Property**

**Description**

TRUE if the "Reserve 4" high limit value of the bar graph object is to be monitored. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties LimitHigh4, ColorLimitHigh4 and TypeLimitHigh4.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "70".

Sub BarGraphLimitConfiguration()
'VBA437
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis to absolute
.TypeLimitHigh4 = False
'Activate monitoring
.CheckLimitHigh4 = True
' set barcolor to "red"
.ColorLimitHigh4 = RGB(255, 0, 0)
'Set upper limit to "70"
.LimitHigh4 = 70
End With
End Sub

See also

TypeLimitHigh4 Property (Page 2453)
LimitHigh4 Property (Page 2307)
ColorLimitHigh4 Property (Page 2186)
BarGraph Object (Page 1914)

CheckLimitHigh5 Property

Description

TRUE if the "Reserve 5" high limit value of the bar graph object is to be monitored. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties LimitHigh5, ColorLimitHigh5 and TypeLimitHigh5.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "80".

Sub BarGraphLimitConfiguration()
'VBA438
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeLimitHigh5 = False
' Activate monitoring
.CheckLimitHigh5 = True
' set barcolor to "black"
.ColorLimitHigh5 = RGB(0, 0, 0)
' Set upper limit to "80"
.LimitHigh5 = 80
End With
End Sub

See also

ColorLimitHigh5 Property (Page 2186)
TypeLimitHigh5 Property (Page 2453)
LimitHigh4 Property (Page 2307)
BarGraph Object (Page 1914)

CheckLimitLow4 Property

Description

TRUE if the "Reserve 4" low limit value of the bar graph object is to be monitored. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties LimitLow4, ColorLimitLow4 and TypeLimitLow4.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "5".

Sub BarGraphLimitConfiguration()
'VBA439
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjec\("Bar1", "HMIBarGraph\"
With objBarGraph
'Set analysis to absolute
.TypeLimitLow4 = False
'Activate monitoring
.CheckLimitLow4 = True
'Set barcolor to "green"
.ColorLimitLow4 = RGB(0, 255, 0)
'Set lower limit to "5"
.LimitLow4 = 5
End With
End Sub

See also

TypeLimitLow4 Property (Page 2454)
LimitLow4 Property (Page 2308)
ColorLimitLow4 Property (Page 2187)
BarGraph Object (Page 1914)

CheckLimitLow5 Property

Description

TRUE if the "Reserve 5" low limit value of the bar graph object is to be monitored. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties LimitLow5, ColorLimitLow5 and TypeLimitLow5.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "0".

Sub BarGraphLimitConfiguration()
    'VBA440
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis to absolute
        .TypeLimitLow5 = False
        'Activate monitoring
        .CheckLimitLow5 = True
        'Set barcolor to "white"
        .ColorLimitLow5 = RGB(255, 255, 255)
        'set lower limit to "0"
        .LimitLow5 = 0
    End With
End Sub

See also

TypeLimitLow5 Property (Page 2455)
LimitLow5 Property (Page 2309)
ColorLimitLow5 Property (Page 2188)
BarGraph Object (Page 1914)

CheckToleranceHigh Property

Description

TRUE if the "Tolerance High" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties ToleranceHigh, ColorToleranceHigh and TypeToleranceHigh.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "45".

Sub BarGraphLimitConfiguration()
    'VBA441
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis to absolute
        .TypeToleranceHigh = False
        'Activate monitoring
        .CheckToleranceHigh = True
        'Set barcolor to "yellow"
        .ColorToleranceHigh = RGB(255, 255, 0)
        'Set upper limit to "45"
        .ToleranceHigh = 45
    End With
End Sub

See also

- TypeToleranceHigh Property (Page 2456)
- ToleranceHigh Property (Page 2442)
- ColorToleranceHigh Property (Page 2189)
- BarGraph Object (Page 1914)

CheckToleranceLow Property

Description

TRUE if the "Tolerance Low" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties ToleranceLow, ColorToleranceLow and TypeToleranceLow.
**Example:**

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "15".

```vba
Sub BarGraphLimitConfiguration()
    'VBA442
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis to absolute
        .TypeToleranceLow = False
        'Activate monitoring
        .CheckToleranceLow = True
        'Set barcolor to "yellow"
        .ColorToleranceLow = RGB(255, 255, 0)
        'Set lower limit to "15"
        .ToleranceLow = 15
    End With
End Sub
```

**See also**

- BarGraph Object (Page 1914)
- TypeToleranceLow Property (Page 2457)
- ToleranceLow Property (Page 2442)
- ColorToleranceLow Property (Page 2190)

**CheckWarningHigh Property**

**Description**

TRUE if the "Warning High" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties WarningHigh, ColorWarningHigh and TypeWarningHigh.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "75".

Sub BarGraphLimitConfiguration()
'VBA443
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjets.AddHMIObjec("Bar1", "HMIBarGraph")
With objBarGraph
  .TypeWarningHigh = False
  .CheckWarningHigh = True
  .ColorWarningHigh = RGB(255, 0, 0)
  .WarningHigh = 75
End With
End Sub

See also

WarningHigh Property (Page 2552)
TypeWarningHigh Property (Page 2458)
ColorWarningHigh Property (Page 2192)
BarGraph Object (Page 1914)

CheckWarningLow Property

Description

TRUE if the "Warning Low" limit value is being monitored for the BarGraph object. BOOLEAN write-read access.

The limit value, the display on reaching the limit value and the type of evaluation are defined via the properties WarningLow, ColorWarningLow and TypeWarningLow.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "12".

```
Sub BarGraphLimitConfiguration()
    'VBA444
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis to absolute
        .TypeWarningLow = False
        'Activate monitoring
        .CheckWarningLow = True
        'Set barcolor to "magenta"
        .ColorWarningLow = RGB(255, 0, 255)
        'Set lower limit to "12"
        .WarningLow = 12
    End With
End Sub
```

See also

- WarningLow Property (Page 2553)
- TypeWarningLow Property (Page 2459)
- ColorWarningLow Property (Page 2193)
- BarGraph Object (Page 1914)

ClearOnError Property

Description

TRUE if the entry in the I/O field is automatically deleted when the input is incorrect. BOOLEAN write-read access.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the I/O field is to be cleared when the input is incorrect:

```vba
Sub IOFieldConfiguration()
'VBA445
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
 .ClearOnError = True
End With
End Sub
```

See also

IOField Object (Page 1992)

ClearOnNew Property

Description

TRUE if the entry in the I/O field is deleted as soon as the I/O field gets the focus. BOOLEAN write-read access.

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the entry in the I/O field is deleted as soon as the field gets the focus:

```vba
Sub IOFieldConfiguration()
'VBA446
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
 .ClearOnNew = True
End With
End Sub
```

See also

IOField Object (Page 1992)
CloseButton Property

Description
TRUE if the ApplicationWindow and PictureWindow objects possess a “Close” button in Runtime. BOOLEAN write-read access.

Example:
The “ApplicationWindowConfig” procedure accesses the properties of the application window. In this example the application window will have a “Close” button in Runtime:

```vba
Sub ApplicationWindowConfig()
  'VBA447
  Dim objAppWindow As HMIApplicationWindow
  Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow1", "HMIApplicationWindow")
  With objAppWindow
    .CloseButton = True
  End With
End Sub
```

See also
- PictureWindow Object (Page 2029)
- ApplicationWindow Object (Page 1911)

CollectValue property

Description
The CollectValue property specifies as an initial value the current status of the active message classes in each case.

- The “Relevant” property has to have the value “TRUE” so that the advanced analog display is taken into account when forming the group display.

ColorAlarmHigh Property

Description
Defines or returns the bar color for the “Alarm High” limit value. LONG write-read access.

- The “CheckAlarmHigh” property must have been set to TRUE if the bar color should change on reaching the limit value.


**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

**Example:**

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "50" and the bar color will change to Red.

```vba
Sub BarGraphLimitConfiguration()
    'VBA449
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .TypeAlarmHigh = False
        .ActivateMonitoring = True
        .CheckAlarmHigh = True
        .ColorAlarmHigh = RGB(255, 0, 0)
        .AlarmHigh = 50
    End With
End Sub
```

**See also**

CheckAlarmHigh Property (Page 2170)
BarGraph Object (Page 1914)

**ColorAlarmLow Property**

**Description**

Defines or returns the bar color for the "Alarm Low" limit value. LONG write-read access.

The "CheckAlarmLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "10" and the bar color will change to Red.

Sub BarGraphLimitConfiguration()
  'VBA450
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis to absolute
    .TypeAlarmLow = False
    'Activate monitoring
    .CheckAlarmLow = True
    'Set barcolor to "red"
    .ColorAlarmLow = RGB(255, 0, 0)
    'Set lower limit to "10"
    .AlarmLow = 10
  End With
End Sub

See also

CheckAlarmLow Property (Page 2171)
BarGraph Object (Page 1914)

ColorBottom Property

Description

Defines or returns the color for the bottom/right stop of the slider object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the color for the lower/right view will be set to "Red":

Sub SliderConfiguration()
'VBA451
Dim objSlider As HMISlider
Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
With objSlider
  .ColorBottom = RGB(255, 0, 0)
End With
End Sub

See also

Slider object (Page 2062)

ColorChangeType Property

Description

TRUE if a color change in the BarGraph object (for instance when a limit value is reached) is to take place segment by segment. If set to FALSE, it defines the change of color for the entire bar. BOOLEAN write-read access.

Example:

The "BarGraphLimitConfiguration()" procedure configures In this example the color change will apply to the whole bar:

Sub BarGraphLimitConfiguration()
'VBA452
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  .ColorChangeType = False
End With
End Sub

See also

BarGraph Object (Page 1914)
ColorLimitHigh4 Property

Description
Defines or returns the color for the "Reserve 4" upper limit value. LONG write-read access. The "CheckLimitHigh4" property must have been set to TRUE if the bar color should change on reaching the limit value.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "70" and the bar color will change to Red.

Sub BarGraphLimitConfiguration()
'VBA453
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeLimitHigh4 = False
' Activate monitoring
.CheckLimitHigh4 = True
' Set barcolor to "red"
.ColorLimitHigh4 = RGB(255, 0, 0)
' Set upper limit to "70"
.LimitHigh4 = 70
End With
End Sub

See also
CheckLimitHigh4 Property (Page 2172)
BarGraph Object (Page 1914)

ColorLimitHigh5 Property

Description
Defines or returns the color for the "Reserve 5" upper limit value. LONG write-read access.
The "CheckLimitHigh5" property must have been set to TRUE if the bar color should change on reaching the limit value.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

**Example:**

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "80" and the bar color will change to "Black".

```vba
Sub BarGraphLimitConfiguration()
'VBA454
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis to absolute
  .TypeLimitHigh5 = False
  'Activate monitoring
  .CheckLimitHigh5 = True
  'Set barcolor to "black"
  .ColorLimitHigh5 = RGB(0, 0, 0)
  'Set upper limit to "80"
  .LimitHigh5 = 80
End With
End Sub
```

See also

- CheckLimitHigh5 Property (Page 2173)
- BarGraph Object (Page 1914)

**ColorLimitLow4 Property**

**Description**

Defines or returns the color for the "Reserve 4" lower limit value. LONG write-read access.

The "CheckLimitLow4" property must have been set to TRUE if the bar color should change on reaching the limit value.
Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "5" and the bar color will change to "Green".

Sub BarGraphLimitConfiguration()
'VBA455
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeLimitLow4 = False
' Activate monitoring
.CheckLimitLow4 = True
' Set barcolor to "green"
.ColorLimitLow4 = RGB(0, 255, 0)
' Set lower limit to "5"
.LimitLow4 = 5
End With
End Sub

See also
CheckLimitLow4 Property (Page 2174)
BarGraph Object (Page 1914)

ColorLimitLow5 Property

Description
Defines or returns the color for the "Reserve 5" lower limit value. LONG write-read access.
The "CheckLimitLow5" property must have been set to TRUE if the bar color should change on reaching the limit value.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "0" and the bar color will change to "White".

```vba
Sub BarGraphLimitConfiguration()
    'VBA456
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis to absolute
        .TypeLimitLow5 = False
        'Activate monitoring
        .CheckLimitLow5 = True
        'Set barcolor to "white"
        .ColorLimitLow5 = RGB(255, 255, 255)
        'Set lower limit to "0"
        .LimitLow5 = 0
    End With
End Sub
```

See also

- CheckLimitLow5 Property (Page 2175)
- BarGraph Object (Page 1914)

ColorToleranceHigh Property

Description

Defines or returns the color for the "Tolerance High" high limit value. LONG write-read access.

The "CheckToleranceHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "45" and the bar color will change to "Yellow".

Sub BarGraphLimitConfiguration()
'VBA457
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis to absolute
  .TypeToleranceHigh = False
  'Activate monitoring
  .CheckToleranceHigh = True
  'Set barcolor to "yellow"
  .ColorToleranceHigh = RGB(255, 255, 0)
  'Set upper limit to "45"
  .ToleranceHigh = 45
End With
End Sub

See also

CheckToleranceHigh Property (Page 2176)
BarGraph Object (Page 1914)

ColorToleranceLow Property

Description

Defines or returns the color for the "Tolerance Low" low limit value. LONG write-read access.

The "CheckToleranceLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "12" and the bar color will change to "Yellow".

Sub BarGraphLimitConfiguration()
'VBA458
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeToleranceLow = False
' Activate monitoring
.CheckToleranceLow = True
'Set barcolor to "yellow"
.ColorToleranceLow = RGB(255, 255, 0)
'Set lower limit to "15"
.ToleranceLow = 15
End With
End Sub

See also

CheckToleranceLow Property (Page 2177)
BarGraph Object (Page 1914)

ColorTop Property

Description

Defines or returns the color for the top/left stop of the slider object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
**Example:**

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the color for the upper/left view will be set to "Orange":

```vba
Sub SliderConfiguration()
    'VBA459
    Dim objSlider As HMIObject
    Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMIObject")
    With objSlider
        .ColorTop = RGB(255, 128, 0)
    End With
End Sub
```

**See also**

Slider object (Page 2062)

**ColorWarningHigh Property**

**Description**

Defines or returns the color for the "Warning High" high limit value. LONG write-read access.

The "CheckWarningHigh" property must have been set to TRUE if the bar color should change on reaching the limit value.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "75" and the bar color will change to "Red".

Sub BarGraphLimitConfiguration()
'VBA460
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObj ect("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeWarningHigh = False
' Activate monitoring
.CheckWarningHigh = True
' Set bar color to "red"
.ColorWarningHigh = RGB(255, 0, 0)
' Set upper limit to "75"
.WarningHigh = 75
End With
End Sub

See also

CheckWarningHigh Property (Page 2178)
BarGraph Object (Page 1914)

ColorWarningLow Property

Description

Defines or returns the color for the "Warning Low" low limit value. LONG write-read access.

The "CheckWarningLow" property must have been set to TRUE if the bar color should change on reaching the limit value.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "12" and the bar color will change to "Magenta".

Sub BarGraphLimitConfiguration()
'VBA461
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis to absolute
.TypeWarningLow = False
' Activate monitoring
.CheckWarningLow = True
' Set barcolor to "magenta"
.ColorWarningLow = RGB(255, 0, 255)
' Set lower limit to "12"
.WarningLow = 12
End With
End Sub

See also

CheckWarningLow Property (Page 2179)
BarGraph Object (Page 1914)

CommonVBSCode Property

Description

Defines the higher-level common declaration section of the actions for the active picture or returns it.

The action editor of the Graphics Designer is used to configure actions at events and properties. In the declaration section of the actions, you can declare tags for a process image as well as create functions and procedures. In Runtime, each VBS action can access these tags, functions and procedures if the picture is active.

If you set "CommonVBSCode", the string is copied to the "Event" and "Property" declaration sections in the action editor. Any code there is overwritten. Therefore, set "CommonVBSCode" first before setting the subordinate declaration sections with "CommonVBSEventArea" or "CommonVBSPROPERTYArea".
Example

In the following example, a tag that is common to all picture objects is declared in the active picture. The common declaration section is then output:

Sub DefineTagInActiveDocument
ActiveDocument.CommonVBSCode = "DIM actionIsdone" & vbCrLf
MsgBox ActiveDocument.CommonVBSCode
End Sub

See also

Document Object (Page 1949)

CommonVBSEventArea property

Description

Defines the "Event" declaration section of the actions for the active picture or returns it.

The action editor of the Graphics Designer is used to configure actions, for example, at events. To this purpose, you can declare tags for a process image as well as create functions and procedures in the "Event" declaration section of the actions. In Runtime each VBS action that was configured for an event can access these tags, functions and procedures if the picture is active.

If you set "CommonVBSEventArea", the string is copied to the "Event" declaration section in the action editor. Any code there is overwritten. Therefore, first read the code set, for example with "CommonVBSCode" before you set the declaration section with "CommonVBSEventArea".

Example

In the following example, two tags are declared in the active picture. The "Event" declaration section is the output:

Sub DefineTagInActiveDocument
ActiveDocument.CommonVBSCode = "DIM actionIsdone" & vbCrLf
ActiveDocument.CommonVBSEventArea = ActiveDocument.CommonVBSEventArea & "DIM " & "eventHasOccurred"
MsgBox ActiveDocument.CommonVBSEventArea
End Sub
**CommonVBSPropertyArea property**

**Description**

Defines the "Property" declaration section of the actions for the active picture or returns it. The action editor of the Graphics Designer is used to configure actions for example at properties. To this purpose you can declare tags for a process image as well as create functions and procedures in the "Property" declaration section of the actions. In Runtime each VBS action that was configured for a property can access these tags, functions and procedures if the picture is active.

If you set "CommonVBSPropertyArea", the string is copied to the "Property" declaration section in the action editor. Any code there is overwritten. Therefore, first read the code set, for example with "CommonVBSCode" before you set the declaration section with "CommonVBSPropertyArea".

**Example**

In the following example, two tags are declared in the active picture. The "Property" declaration section is then output:

```vba
Sub DefineTagInActiveDocument
    ActiveDocument.CommonVBSCode = "DIM actionIsdone" & vbCrLf
    ActiveDocument.CommonVBSPropertyArea = ActiveDocument.CommonVBSPropertyArea & "DIM propertyIsChanged"
    MsgBox ActiveDocument.CommonVBSPropertyArea
End Sub
```

**CommandLine Property**

**Description**

Returns the start parameter as a string if the application is opened via Start>Execute "Grafexe.exe start parameter". Read only access.

**Example:**

In this example a message containing the start parameter is output on opening the document.

```vba
Sub Document_Opened(CancelForwarding As Boolean)
    'VBA462
    MsgBox Application.Commandline
End Sub
```
Compiled Property

Description

TRUE if the source code of a C script or VB script was successfully compiled. BOOLEAN read access.

Example:

In the following example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle will enlarge every time you click the button. A VB script will be used for this purpose:

```vba
Sub IncreaseCircleRadiusWithVBScript()
'VBA463
Dim objButton As HMIButton
Dim objCircleA As HMICircle
Dim objEvent As HMIEvent
Dim objVBScript As HMIScriptInfo
Dim strCode As String
strCode = "Dim objCircle" & vbCrLf & "Set objCircle = "
strCode = strCode & "hmiRuntime.ActiveScreen.ScreenItems(""CircleVB"""")"
strCode = strCode & vbCrLf & "objCircle.Radius = objCircle.Radius + 5"
Set objCircleA = ActiveDocument.HMIObjects.AddHMIObject("CircleVB", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircleA
  .Top = 100
  .Left = 100
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 200
  .Text = "Increase Radius"
End With
'On every mouseclick the radius will be increased:
Set objEvent = objButton.Events(1)
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
objVBScript.SourceCode = strCode
Select Case objVBScript.Compiled
Case True
  MsgBox "Compilation OK!"
Case False
  MsgBox "Errors by compilation!"
End Select
End Sub
```
See also

SourceCode Property (Page 2424)
ScriptInfo Object (Page 2058)

ConfigurationFileName Property

Description

Returns the file name and full path of the configuration file for the open project. STRING read access.

Example:

The "ShowConfigurationFileName() procedure outputs the configuration file path for the current picture:

Sub ShowConfigurationFileName()
  'VBA464
  MsgBox ActiveDocument.Application.ConfigurationFileName
End Sub

See also

Application Property (Page 2121)
Application Object (Page 1909)

ConnectionPoints property

Description

Returns the number of connection points of an object.
Example: Number of connection points of a rectangle

In this example, a rectangle is inserted and the number of connection points is output:

```
Sub CountConnectionPoints()
    'VBA229
    Dim objRectangle As HMIRectangle
    Dim objConnPoints As HMIConnectionPoints
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    Set objConnPoints = ActiveDocument.HMIObjects("Rectangle1").ConnectionPoints
    MsgBox "Rectangle1 has " & objConnPoints.Count & " connectionpoints."
End Sub
```

ConnectorObjects property

Description

Only used internally.

See also

- 3DBarGraph Object (Page 1894)
- ActiveXControl Object (Page 1900)
- AdvancedAnalogDisplay object (Page 1901)
- AdvancedStateDisplay object (Page 1905)
- ApplicationWindow Object (Page 1911)
- BarGraph Object (Page 1914)
- Button Object (Page 1921)
- CheckBox Object (Page 1924)
- Circle Object (Page 1928)
- CircularArc Object (Page 1931)
- ComboBox object (Page 1934)
- CustomizedObject Object (Page 1938)
- DataSetObj object (Page 1943)
- DotNetControl object (Page 1954)
- Ellipse Object (Page 1957)
- EllipseArc Object (Page 1960)
- EllipseSegment Object (Page 1963)
- FaceplateObject object (Page 1969)
- GraphicObject Object (Page 1975)
- Group Object (Page 1978)
ConnectorType property

Description

Defines the type of connector:

<table>
<thead>
<tr>
<th>ConnectorType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Both objects are connected by a polyline made up of horizontal and vertical parts.</td>
</tr>
<tr>
<td>Simple</td>
<td>Both objects are connected by a straight line between the connecting points.</td>
</tr>
</tbody>
</table>
ControlType property

Description
Returns the name range of the control.

CopyPasteSettings property

Description
Only used internally.

See also
Application Object (Page 1909)

CornerRadius property

Description
Defines the rounding radius of the rectangle which enclose objects in the advanced analog display. The values are defined in pixels.

The range of values which can be displayed for the corner radius depends on the values set for the "height" and "width" properties. The maximum corner radius value which can be displayed is equivalent to 50% of the lower one of the "height" or "width" values. The maximum value is used if higher values are entered.

Count Property

Description
Returns the number of elements in the specified listing. LONG read access
Example:

In the following example a new picture will be created and a pair of objects will be inserted. The number of inserted objects will be output at the end:

Sub ObjectsInActiveDocument()
'VBA465
Dim objCircle As HMIObject
Dim objRectangle As HMIObject
Dim objDocument As Document
Set objDocument = Application.Documents.Add(hmiOpenDocumentTypeVisible)
Dim iIndex As Integer
iIndex = 1
For iIndex = 1 To 5
Set objCircle = objDocument.HMIObjects.AddHMIObject("Circle" & iIndex, "HMIObject")
Set objRectangle = objDocument.HMIObjects.AddHMIObject("Rectangle" & iIndex, "HMIObject")
With objCircle
 .Top = (10 * iIndex)
 .Left = (10 * iIndex)
End With
With objRectangle
 .Top = ((10 * iIndex) + 50)
 .Left = (10 * iIndex)
End With
Next iIndex
MsgBox "There are " & objDocument.HMIObjects.Count & " objects in the document"
End Sub

See also

- VariableTriggers Object (Listing) (Page 2101)
- Views Object (Listing) (Page 2103)
- VariableStateValues Object (Listing) (Page 2097)
- ToolbarItems Object (Listing) (Page 2085)
- Toolbars Object (Listing) (Page 2081)
- SymbolLibraries Object (Listing) (Page 2073)
- SelectedObjects object (Listing) (Page 2060)
- Properties Object (Listing) (Page 2041)
- HMIObjects Object (Listing) (Page 1990)
- MenuItems Object (Listing) (Page 2015)
- Menus Object (Listing) (Page 2011)
- Layers Object (Listing) (Page 2003)
- LanguageTexts Object (Listing) (Page 2000)
- LanguageFonts Object (Listing) (Page 1997)
CQBackColorOff..ColorOn property

Description

Specifies for the selected message type and the state "Came In Acknowledged" which color the background of the value to be displayed assumes for flashing status "Off" (CBackColorOff) or "On" (CBackColorOn) when the arrival of a message is acknowledged.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

CQBackFlash property

Description

Specifies for the selected message type and status "Came In Acknowledged" whether the background of the value to be displayed flashes when the arrival of a message is acknowledged.

CQTextColorOff..ColorOn property

Description

Specifies for the selected message type and the state "Came In Acknowledged" which color the text of the value to be displayed assumes for flashing status "Off" (CTextColorOff) or "On" (CTextColorOn) when the arrival of a message is acknowledged.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

CQTextFlash property

Description
Specifies for the selected message type and status "Came In Acknowledged" whether the background of the text to be displayed flashes when the arrival of a message is acknowledged.

CTextColorOff..ColorOn property

Description
Specifies for the selected message type and the state "Came In" which color the text of the value to be displayed assumes for flashing status "Off" (CTextColorOff) or "On" (CTextColorOn).

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

CTextFlash property

Description
Specifies for the selected message type and status "Came In" whether the text of the value to be displayed flashes when a message is received.

CurrentDataLanguage Property

Description
Defines the project language or returns the language identifier as a decimal value. LONG read-write access.
Example:
The "ShowDataLanguage()" procedure outputs the currently set project language:

```vba
Sub ShowDataLanguage()
    'VBA466
    MsgBox Application.CurrentDataLanguage
End Sub
```

See also
- Application Property (Page 2121)
- DataLanguageChanged Event (Page 1762)
- Language-Dependent Configuration with VBA (Page 1624)

**CurrentDesktopLanguage Property**

**Description**
Returns the language identifier of the currently set user interface language as a decimal value. LONG read access.

**Example:**
The "ShowDesktopLanguage()" procedure outputs the currently set user interface language:

```vba
Sub ShowDesktopLanguage()
    'VBA467
    MsgBox Application.CurrentDesktopLanguage
End Sub
```

See also
- Application Property (Page 2121)
- Application Object (Page 1909)
- DesktopLanguageChanged event (Page 1763)
- Language-Dependent Configuration with VBA (Page 1624)
CursorControl Property

Description
TRUE, when Alpha Cursor mode is activated, the cursor skips to the next field in the TAB sequence after exiting the field. BOOLEAN write-read access.
The CursorMode property must be set to TRUE.

Example:
The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the cursor will skip into the next field when another field is exited. For this to work, the Cursor mode property must first be set to TRUE.

Sub IOFieldConfiguration()
  'VBA468
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIOObjects.AddHMIObject("IOField1", "HMIIOField")
  Application.ActiveDocument.CursorMode = True
  With objIOField
    .CursorControl = True
  End With
End Sub

See also
TabOrderAlpha Property (Page 2429)
TabOrderSwitch Property (Page 2433)
CursorMode Property (Page 2206)
ActiveDocument Property (Page 2107)
TextList Object (Page 2076)
IOField Object (Page 1992)

CursorMode Property

Description
TRUE if the "Alpha Cursor" mode is to be activated. FALSE if the "Tab order" mode is to be activated. BOOLEAN write-read access.
Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the "Alpha Cursor" mode will be activated:

Sub ActiveDocumentConfiguration()
    'VBA469
    Application.ActiveDocument.CursorMode = True
End Sub

See also

CursorControl Property (Page 2206)
ActiveDocument Property (Page 2107)
Documents Object (Listing) (Page 1952)

CustomMenus Property

Description

Returns a listing of the available user-defined menus.

Example:

The "ShowCustomMenuInformation()" procedure outputs the Key and Label of all user-defined menus in the current picture:

Sub ShowCustomMenuInformation()
    'VBA470
    Dim strKey As String
    Dim strLabel As String
    Dim strOutput As String
    Dim iIndex As Integer
    For iIndex = 1 To ActiveDocument.CustomMenus.Count
        strKey = ActiveDocument.CustomMenus(iIndex).Key
        strLabel = ActiveDocument.CustomMenus(iIndex).Label
        strOutput = strOutput & vbCrLf & "Key: " & strKey & "  Label: " & strLabel
    Next iIndex
    If 0 = ActiveDocument.CustomMenus.Count Then
        strOutput = "There are no custommenus for the document created."
    End If
    MsgBox strOutput
End Sub
CustomToolbars Property

Description

Returns a listing of the available user-defined toolbars.

Example:

The "ShowCustomToolbarInformation()" procedure outputs the Key values of all user-defined toolbars in the current picture:

```vba
Sub ShowCustomToolbarInformation()
    'VBA471
    Dim strKey As String
    Dim strOutput As String
    Dim iIndex As Integer
    For iIndex = 1 To ActiveDocument.CustomToolbars.Count
        strKey = ActiveDocument.CustomToolbars(iIndex).Key
        strOutput = strOutput & vbCrLf & "Key: " & strKey
    Next iIndex
    If 0 = ActiveDocument.CustomToolbars.Count Then
        strOutput = "There are no toolbars created for this document."
    End If
    MsgBox strOutput
End Sub
```

See also

Application Property (Page 2121)
ActiveDocument Property (Page 2107)
Menu Object (Page 2009)

CycleName Property

Description

Returns the name of the specified tag trigger. Read only access.
Example:

See also

VariableTrigger Object (Page 2099)

CycleTime Property

Description

Returns the cycle time of the specified tag trigger. Read only access.

Example:

See also

VariableTrigger Object (Page 2099)

CycleType Property

Description

Defines or returns the cycle type.

Example:

The "DynamicToRadiusOfNewCircle(hmiCircle As IHMICircle)" procedure creates a dynamic for the radius of a circle. In this example the radius of the circle will be set every two seconds:

```vba
Sub DynamicToRadiusOfNewCircle()
    'VBA474
    Dim objCircle As hmiCircle
    Dim VariableTrigger As HMIVariableTrigger
    Set objCircle = Application.ActiveDocument.HMIObjects.AddHMIObject("Circle1", "HMICircle")
    Set VariableTrigger = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVariableDirect, "NewDynamic1")
    VariableTrigger.CycleType = hmiVariableCycleType_2s
End Sub
```
See also

VariableTrigger Object (Page 2099)
Configuring Dynamics in the Properties of Pictures and Objects (Page 1687)

D

DataFormat Property

Description

Defines or returns the data type of the IOField object. Value range from 0 to 3.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>0</td>
</tr>
<tr>
<td>Decimal</td>
<td>1</td>
</tr>
<tr>
<td>String</td>
<td>2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>3</td>
</tr>
</tbody>
</table>

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example, data type "Decimal" will be set for the I/O field:

```vba
Sub IOFieldConfiguration()
  'VBA475
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .DataFormat = 1
  End With
End Sub
```

See also

IOField Object (Page 1992)

DefaultHMIObjects Property

Description

Returns the HMIObjects listing.
Example:

The "ShowDefaultObjectNames()" procedure outputs all the object names contained in the HMIDefaultObjects listing:

```vba
Sub ShowDefaultObjectNames()
  'VBA476
  Dim strOutput As String
  Dim iIndex As Integer
  For iIndex = 1 To Application.DefaultHMIObjects.Count
    strOutput = strOutput & vbCrLf & Application.DefaultHMIObjects(iIndex).ObjectName
  Next iIndex
  MsgBox strOutput
End Sub
```

See also

HMIDefaultObjects Object (Listing) (Page 1985)

DestinationLink Property

Description

Returns the Destination object. Use the DestinationLink property to configure the destination object in the case of a direct connection.
Example:

Use the DestinationLink property to return the DestLink object. In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

```vba
Sub DirectConnection()
    'VBA477
    Dim objButton As HMIButton
    Dim objRectangleA As HMIRectangle
    Dim objRectangleB As HMIRectangle
    Dim objEvent As HMIEvent
    Dim objDirConnection As HMIDirectConnection
    Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
    Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    With objRectangleA
        .Top = 100
        .Left = 100
    End With
    With objRectangleB
        .Top = 250
        .Left = 400
        .BackColor = RGB(255, 0, 0)
    End With
    With objButton
        .Top = 10
        .Left = 10
        .Width = 100
        .Text = "SetPosition"
    End With
    'Directconnection is initiated by mouseclick:
    Set objDirConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
    With objDirConnection
        'Sourceobject: Property "Top" of Rectangle_A
        .SourceLink.Type = hmiSourceTypeProperty
        .SourceLink.ObjectName = "Rectangle_A"
        .SourceLink.AutomationName = "Top"
        'Targetobject: Property "Left" of Rectangle_B
        .DestinationLink.Type = hmiDestTypeProperty
        .DestinationLink.ObjectName = "Rectangle_B"
        .DestinationLink.AutomationName = "Left"
    End With
End Sub
```

See also

AutomationName Property (Page 2124)
ObjectName Property (Page 2347)
Type Property (Page 2450)
DirectConnection Object (Page 1947)

Direction Property

Description
Defines or returns the bar direction. BOOLEAN write-read access.

Slider
Defines or returns the position of the Slider object. BOOLEAN write-read access.

<table>
<thead>
<tr>
<th>Position/Bar Axis</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical/Negative</td>
<td>TRUE</td>
</tr>
<tr>
<td>Horizontal/Positive</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Example:
The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the position of the Slider object will be set to "Vertical":

Sub SliderConfiguration()
  'VBA478
  Dim objSlider As HMISlider
  Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
  With objSlider
    .Direction = True
  End With
End Sub

See also
Slider object (Page 2062)
3DBarGraph Object (Page 1894)

DisablePerformanceWarnings property

Description
Only used internally.

See also
Application Object (Page 1909)
DisableVBAEvents Property

Description
TRUE if Event Handling is disabled. BOOLEAN write-read access.

Example:
The "DisableVBAEvents()" procedure disables Event Handling:

```vba
Sub DisableVBAEvents()
    Application.DisableVBAEvents = False
End Sub
```

See also
Application Object (Page 1909)
Event Handling (Page 1709)

Display property

Description
Only used internally.

See also
ObjConnection object (Page 2019)

DisplayName Property

Description
Returns the name of the property attribute. STRING read access.
Thus the expression "MsgBox ActiveDocument.HMIObjects("Circle_1").Properties("Height").DisplayName" would output the result "Height".
Example:

The "ShowAllObjectDisplayNames()" procedure outputs all the property attribute names of standard objects contained in the message box:

```vba
Sub ShowAllObjectDisplayNames()
    'VBA480
    Dim strOutput As String
    Dim iIndex1 As Integer
    iIndex1 = 1
    strOutput = "List of all properties-displaynames from object " & Application.DefaultHMIObjects(1).ObjectName & "" & vbCrLf & vbCrLf
    For iIndex1 = 1 To Application.DefaultHMIObjects(1).Properties.Count
        strOutput = strOutput & Application.DefaultHMIObjects(1).Properties(iIndex1).DisplayName & " / " & vbCrLf
    Next iIndex1
    MsgBox strOutput
End Sub
```

See also

Property Object (Page 2042)

DisplayOptions Property

Description

Defines the assignment of the "Button" or "Round button" object or returns its value. Value range from 0 to 3.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic or text</td>
<td>0</td>
</tr>
<tr>
<td>Graphic and text</td>
<td>1</td>
</tr>
<tr>
<td>Text only</td>
<td>2</td>
</tr>
<tr>
<td>Graphic only</td>
<td>3</td>
</tr>
</tbody>
</table>

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button.
In this example the button is assigned "Graphic and text":

Sub ButtonConfiguration()
  'VBA814
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
  With objButton
    .DisplayOptions = 1
  End With
End Sub

See also

Button Object (Page 1921)

DisplayText Property

Description

Returns the value for the "Label" or "TooltipText" property of the following objects (STRING read access):

- Menu Object
- MenuItem Object
- ToolbarItem Object

Example:

The "ShowLabelTexts()" procedure outputs all the labels of the first user-defined menu in the current picture:

Sub ShowLabelTexts()
  'VBA481
  Dim objLangText As HMILanguageText
  Dim iIndex As Integer
  For iIndex = 1 To ActiveDocument.CustomMenus(1).LDLabelTexts.Count
    Set objLangText = ActiveDocument.CustomMenus(1).LDLabelTexts(iIndex)
    MsgBox objLangText.DisplayName
  Next iIndex
End Sub
Documents Property

Description

Returns the Documents listing containing all open pictures. The open pictures are in chronological order.

Example:

In the following example the names of all open pictures are output:

Sub ShowDocuments()
    'VBA482
    Dim colDocuments As Documents
    Dim objDocument As Document
    Dim strOutput As String
    Set colDocuments = Application.Documents
    strOutput = "List of all opened documents:" & vbCrLf
    For Each objDocument In colDocuments
        strOutput = strOutput & vbCrLf & objDocument.Name
    Next objDocument
    MsgBox strOutput
End Sub

See also

Application Property (Page 2121)
Application Object (Page 1909)
**DrawInsideFrame property**

**Description**

Defines for all line thicknesses greater than "1" whether the border lines are to be drawn inside the object frame or symmetrically on the frame.

<table>
<thead>
<tr>
<th>Yes</th>
<th>The border lines are drawn inside the object frame.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>The border lines are drawn symmetrically on the object frame.</td>
</tr>
</tbody>
</table>

**DropDownListStyle property**

**Description**

Defines whether the entries in the "TextList" object are displayed in a drop-down list box.

**Dynamic Property**

**Description**

Returns the dynamics of a property.

**Example:**

Use the Dynamic property if you wish to return, say, an existing dynamic. In the following example all possibly available object property dynamics are output in the active picture:

```vba
Sub ShowPropertiesDynamicsOfAllObjects()
  'VBA483
  Dim objObject As HMIObject
  Dim colObjects As HMIObjects
  Dim colProperties As HMIProperties
  Dim objProperty As HMIProperty
  Dim strOutput As String
  Set colObjects = Application.ActiveDocument.HMIObjects
  For Each objObject In colObjects
    Set colProperties = objObject.Properties
    For Each objProperty In colProperties
      If 0 <> objProperty.DynamicStateType Then
        strOutput = strOutput & vbCrLf & objObject.ObjectName & " - " & objProperty.DisplayName & ": Statetype " & objProperty.Dynamic.DynamicStateType
      End If
    Next objProperty
  Next objObject
  MsgBox strOutput
End Sub
```
See also

Property Object (Page 2042)

DynamicStateType property

Description

Only used internally.

See also

VBA Reference: ActionDynamic (Page 1747)

EditAtOnce Property

Description

TRUE, if accessing the field with the <TAB> key permits input immediately and without further action. BOOLEAN write-read access.

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example it shall be possible to enter input on skipping into the I/O field:

```vba
Sub IOFieldConfiguration()
  VBA484
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .EditAtOnce = True
  End With
End Sub
```

See also

TextList Object (Page 2076)

IOField Object (Page 1992)
ElseCase Property

Description

Defines or returns the value for the dynamic property outside of the configured value range.

Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and three analog value ranges will be created:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA485
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.Add 50, 40
    .AnalogResultInfos.Add 100, 80
    .AnalogResultInfos.ElseCase = 100
End With
End Sub
```

See also

- AnalogResultInfos Object (Listing) (Page 1908)
- AnalogResultInfo Object (Page 1907)
- Add Method (AnalogResultInfos Listing) (Page 1787)

Enabled Property

Description

TRUE if the menu, the menu entry or the icon is activated and can be selected. Applies only to user-defined menus and toolbars. BOOLEAN write-read access.
Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"): In this example the second menu point in user-defined menu "Delete Objects" is grayed out and cannot be selected in the Graphics Designer:

```
Sub DisableMenuItem()
    'VBA486
    Dim objMenu As HMI.Menu
    Dim objMenuItem As HMI.MenuItem

    'Add a new menu "Delete objects"
    Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")
    'Add two menuitems to the new menu
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete rectangles")
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete circles")
    'Disable menuitem "Delete circles"
    With ActiveDocument.CustomMenus("DeleteObjects").MenuItems("DeleteAllCircles")
        .Enabled = False
    End With
End Sub
```

See also

- ToolbarItem Object (Page 2083)
- MenuItem Object (Page 2013)
- Menu Object (Page 2009)
- Configuring Menus and Toolbars (Page 1626)

EnableFlashing property

Description

Specifies whether the value for status "OK" and "Simulation" appears flashing or not in the advanced analog display in Runtime.

For the flashing to be visible in Runtime, the font flashing color must be different to the background flashing color.
EndAngle Property

Description
Defines or returns the end of the object for the CircularArc, EllipseArc, EllipseSegment and PieSegment objects. The information is in counterclockwise direction in degrees, beginning at the 12:00 clock position.

Example:
The "PieSegmentConfiguration()" procedure accesses the properties of the Pie Segment. In this example the pie segment begins at 40° and ends at 180°:

```vba
Sub PieSegmentConfiguration()
'VBA487
Dim objPieSegment As HMIPieSegment
Set objPieSegment = ActiveDocument.HMIObjects.AddHMIObj("PieSegment1", "HMIPieSegment")
With objPieSegment
    .StartAngle = 40
    .EndAngle = 180
End With
End Sub
```

See also
- StartAngle Property (Page 2426)
- PieSegment Object (Page 2032)
- EllipseSegment Object (Page 1963)
- EllipseArc Object (Page 1960)
- CircularArc Object (Page 1931)

EventQuitMask property

Description
The events "Operator request" and "Measuring point blocked" are not acknowledgeable events in the PCS 7 environment. Using the "@EventQuit" tag and the "EventQuitMask" property in Runtime, these events are automatically indicated as acknowledged to prevent flashing during the calculation of the group displays. The start value of the attribute is then 0x00000011 (17). The value of the "EventQuitMask" property should be identical for all group display objects, advanced analog display and advanced status display, and for the "@EventQuit" tag.

By setting further acknowledgment bits, you can indicate other events as being acknowledged as well with the display of the group display object and the advanced analog and status display.
Events Property

Description

Returns the Events listing. Use the Events property to define the event that will trigger an action. Use the index number to define the event that is intended to be configured:

- You configure an action on a property with VBA by using the "Events(9)" property, where the index "1" stands for the event "Upon change";
- To configure an action onto an object with the aid of VBA, use the "Events(Index)" property, where "Index" stands for the trigger event (see table):

<table>
<thead>
<tr>
<th>Index</th>
<th>EventType (depending upon the object used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hmiEventTypeNotDefined</td>
</tr>
<tr>
<td>1</td>
<td>hmiEventTypeMouseClick</td>
</tr>
<tr>
<td>2</td>
<td>hmiEventTypeMouseLButtonDown</td>
</tr>
<tr>
<td>3</td>
<td>hmiEventTypeMouseLButtonUp</td>
</tr>
<tr>
<td>4</td>
<td>hmiEventTypeMouseRButtonDown</td>
</tr>
<tr>
<td>5</td>
<td>hmiEventTypeMouseRButtonUp</td>
</tr>
<tr>
<td>6</td>
<td>hmiEventTypeKeyboardDown</td>
</tr>
<tr>
<td>7</td>
<td>hmiEventTypeKeyDown</td>
</tr>
<tr>
<td>8</td>
<td>hmiEventTypeKeyUp</td>
</tr>
<tr>
<td>9</td>
<td>hmiEventTypeObjectChange</td>
</tr>
<tr>
<td>10</td>
<td>hmiEventTypeOpenPicture</td>
</tr>
<tr>
<td>11</td>
<td>hmiEventTypePictureOpen</td>
</tr>
<tr>
<td>12</td>
<td>hmiEventTypePictureClose</td>
</tr>
<tr>
<td>13</td>
<td>hmiEventTypeObjectDefined</td>
</tr>
<tr>
<td>14</td>
<td>hmiEventTypeFocusEnter</td>
</tr>
<tr>
<td>15</td>
<td>hmiEventTypeLastTriggerType</td>
</tr>
<tr>
<td>16</td>
<td>hmiEventTypeObjSpecificTriggerStart</td>
</tr>
</tbody>
</table>
Example:

In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
'VBA488
Dim objButton As HMIButton
Dim objRectangleA As HMIRectangle
Dim objRectangleB As HMIRectangle
Dim objEvent As HMIEvent
Dim objDirConnection As HMIDirectConnection
Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objRectangleA
  .Top = 100
  .Left = 100
End With
With objRectangleB
  .Top = 250
  .Left = 400
  .BackColor = RGB(255, 0, 0)
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 100
  .Text = "SetPosition"
End With
'
'Directconnection is initiated by mouseclick:
Set objDirConnection =
objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
With objDirConnection
' Sourceobject: Property "Top" of Rectangle_A
  .SourceLink.Type = hmiSourceTypeProperty
  .SourceLink.ObjectName = "Rectangle_A"
  .SourceLink.AutomationName = "Top"
',
'Targetobject: Property "Left" of Rectangle_B
  .DestinationLink.Type = hmiDestTypeProperty
  .DestinationLink.ObjectName = "Rectangle_B"
  .DestinationLink.AutomationName = "Left"
End With
End Sub

See also

Events Object (Listing) (Page 1967)
Configuring Event-Driven Actions with VBA (Page 1699)
EventName property

Description
Returns the name of the "Event" object.

Example
In this example the event names and event types of all objects in the active pictures are put out. In order for this example to work, insert some objects into the active picture and configure different events.

```vba
Sub ShowEventsOfAllObjectsInActiveDocument()
  'VBA252
  Dim colEvents As HMIEvents
  Dim objEvent As HMIEvent
  Dim iMax As Integer
  Dim iIndex As Integer
  Dim iAnswer As Integer
  Dim strEventName As String
  Dim strObjectName As String
  Dim varEventType As Variant
  iIndex = 1
  iMax = ActiveDocument.HMIObjects.Count
  For iIndex = 1 To iMax
    Set colEvents = ActiveDocument.HMIObjects(iIndex).Events
    strObjectName = ActiveDocument.HMIObjects(iIndex).ObjectName
    For Each objEvent In colEvents
      strEventName = objEvent.EventName
      varEventType = objEvent.EventType
      iAnswer = MsgBox("Objectname: " & strObjectName & vbCrLf & "Eventtype: " & varEventType & vbCrLf & "Eventname: " & strEventName, vbOKCancel)
      If vbCancel = iAnswer Then Exit For
    Next objEvent
  Next iIndex
End Sub
```

EventType Property

Description
Returns the event type that is configured on the specified object.

<table>
<thead>
<tr>
<th>Index</th>
<th>EventType (depending upon the object used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hmiEventTypeNotDefined</td>
</tr>
<tr>
<td>1</td>
<td>hmiEventTypeMouseClick</td>
</tr>
<tr>
<td>2</td>
<td>hmiEventTypeMouseLButtonDown</td>
</tr>
<tr>
<td>3</td>
<td>hmiEventTypeMouseLButtonUp</td>
</tr>
<tr>
<td>Index</td>
<td>EventType (depending upon the object used)</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>hmiEventTypeMouseRButtonDown</td>
</tr>
<tr>
<td>5</td>
<td>hmiEventTypeMouseRButtonUp</td>
</tr>
<tr>
<td>6</td>
<td>hmiEventTypeKeyboardDown</td>
</tr>
<tr>
<td>7</td>
<td>hmiEventTypeKeyboardUp</td>
</tr>
<tr>
<td>8</td>
<td>hmiEventTypeFocusEnter</td>
</tr>
<tr>
<td>9</td>
<td>hmiEventTypeObjectChange</td>
</tr>
<tr>
<td>10</td>
<td>hmiEventTypeOpenPicture</td>
</tr>
<tr>
<td>11</td>
<td>hmiEventTypePictureOpen</td>
</tr>
<tr>
<td>12</td>
<td>hmiEventTypePictureClose</td>
</tr>
<tr>
<td>13</td>
<td>hmiEventTypeObjectDefined</td>
</tr>
<tr>
<td>14</td>
<td>hmiEventTypeFocusEnter</td>
</tr>
<tr>
<td>15</td>
<td>hmiEventTypeLastTriggerType</td>
</tr>
<tr>
<td>16</td>
<td>hmiEventTypeObjSpecificTriggerStart</td>
</tr>
</tbody>
</table>

**Example:**

Use the EventType property to edit a previously configured event. In the following example the event "Mouse Action" will be configured, but then changed to "Pressed":

```vba
Sub AddActionToObjectTypeCScript()
'VBA489
Dim objEvent As HMIEvent
Dim objCScript As HMIScriptInfo
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_AB", "HMICircle")
' 
'C-action is initiated by click on object circle
Set objEvent = objCircle.Events(1)
Set objCScript = objEvent.Actions.AddAction(hmiActionCreationTypeCScript)
MsgBox "the type of the projected event is " & objEvent.EventType
End Sub
```

**See also**

- Events Object (Listing) (Page 1967)
- Configuring Event-Driven Actions with VBA (Page 1699)
**Exponent Property**

**Description**

TRUE if numbers are to be displayed on the BarGraph object using exponents (e.g. "1.00e+000"). BOOLEAN write-read access.

**Example:**

The "BarGraphConfiguration()" procedure configures In this example numbers are to be displayed on the bar using exponents:

```vba
Sub BarGraphConfiguration()
    'VBA490
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .Exponent = True
    End With
End Sub
```

**See also**

BarGraph Object (Page 1914)

**ExtendedOperation Property**

**Description**

TRUE if the slider on the Slider object is set to the associated end value (minimum value/maximum value). This is done by clicking the mouse in an area outside the current regulator setting. BOOLEAN write-read access.

**Example:**

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the ExtendedOperation property will be set to TRUE:

```vba
Sub SliderConfiguration()
    'VBA491
    Dim objSlider As HMISlider
    Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
    With objSlider
        .ExtendedOperation = True
    End With
End Sub
```
See also

Slider object (Page 2062)

ExtendedZoomingEnable Property

Description

TRUE, if the selected process picture in Runtime may be zoomed in or out using the mouse wheel. This happens by pushing the <CTRL> key while the mouse wheel is turned. If the mouse wheel is turned away from the palm of the hand, the zoom factor increases.

BOOLEAN write-read access.

Requirements for using the zoom function:

- Mouse driver by Logitech or Microsoft Intellimouse
- Mouse wheel must be set to "Autoscroll".
- In the computer properties, the "Graphics Runtime" tab control must have the "Extended zooming" function enabled for all process pictures.

Example:

The procedure "DocConfiguration()" accesses picture properties.

In this example, the property ExtendedZoomingEnable is set to TRUE:

```vba
Sub DocConfiguration()
'VBA815
Dim objDoc As Document
Set objDoc = ActiveDocument
With objDoc
 .ExtendedZoomingEnable = True
End With
End Sub
```

FaceplateType property

Description

Sets the faceplate type of the faceplate instance and returns its name. The faceplate type is "Const" and can therefore only be set once.
Usage

Use the Add method to create a new "faceplate instance" object in a picture. "Properties.Item(3)" is used to access the FaceplateType property:

Sub FaceplateInstance_and_Properties()
'VBA847
Dim objFaceplateInstance As HMIFaceplateObject
Set objFaceplateInstance = ActiveDocument.HMIObjects.AddHMIObject("faceplate instance", "HMIFaceplateObject")
objFaceplateInstance.Properties.Item(3).value = "Faceplate1.fpt"
MsgBox "Faceplate """" & objFaceplateInstance.Properties.Item(3).value & """ is used."
End Sub

Family Property

Description

Defines or returns the language-dependent font.
The following example sets the font attributes of a button for French and English:

```vba
Sub ExampleForLanguageFonts()
    'VBA492
    Dim colLangFonts As HMILanguageFonts
    Dim objButton As HMIButton
    Dim iStartLangID As Integer
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    iStartLangID = Application.CurrentDataLanguage
    With objButton
        .Text = "Command"
        .Width = 100
    End With
    Set colLangFonts = objButton.LDFonts
    'To do typesettings for french:
    With colLangFonts.ItemByLCID(1036)
        .Family = "Courier New"
        .Bold = True
        .Italic = False
        .Underlined = True
        .Size = 12
    End With
    'To do typesettings for english:
    With colLangFonts.ItemByLCID(1033)
        .Family = "Times New Roman"
        .Bold = False
        .Italic = True
        .Underlined = False
        .Size = 14
    End With
    Application.CurrentDataLanguage = 1036
    .Text = "Command"
    MsgBox "Datalanguage is changed in french"
    Application.CurrentDataLanguage = 1033
    .Text = "Command"
    MsgBox "Datalanguage is changed in english"
    Application.CurrentDataLanguage = iStartLangID
    MsgBox "Datalanguage is changed back to startlanguage."
End With
End Sub
```

See also

- Underlined Property (Page 2460)
- Size Property (Page 2419)
- Parent Property (Page 2359)
FillBackColor property

Description

Only used internally.

See also

Document Object (Page 1949)

FillColor Property

Description

Defines or returns the fill pattern color for the object. LONG read-write access.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

**Example:**

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the background color will be set to "Yellow".

```vba
Sub RectangleConfiguration()
'VBA493
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIOObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
  .FillColor = RGB(255, 255, 0)  'Yellow
End With
End Sub
```
See also

- Button Object (Page 1921)
- StaticText Object (Page 2067)
- Slider object (Page 2062)
- TextList Object (Page 2076)
- RoundRectangle Object (Page 2055)
- RoundButton Object (Page 2052)
- Rectangle Object (Page 2049)
- Polygon Object (Page 2035)
- PieSegment Object (Page 2032)
- OptionGroup Object (Page 2025)
- GroupDisplay Object (Page 1980)
- GraphicObject Object (Page 1975)
- IOField Object (Page 1992)
- EllipseSegment Object (Page 1963)
- Ellipse Object (Page 1957)
- Document Object (Page 1949)
- Circle Object (Page 1928)
- CheckBox Object (Page 1924)
- BarGraph Object (Page 1914)
- 3DBarGraph Object (Page 1894)

Filling Property

Description

TRUE if an object with closed frame lines (such as a Circle or Rectangle) can be filled (as in the fill level of a tank, for example). BOOLEAN write-read access.

To set the fill level of the object, use the FillingIndex property.
Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example a rectangle can be used to display the fill level:

Sub RectangleConfiguration()
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .Filling = True
    End With
End Sub

See also

- FillingIndex Property (Page 2233)
- StaticText Object (Page 2067)
- Slider object (Page 2062)
- RoundRectangle Object (Page 2055)
- RoundButton Object (Page 2052)
- Rectangle Object (Page 2049)
- Polygon Object (Page 2035)
- PieSegment Object (Page 2032)
- OptionGroup Object (Page 2025)
- GraphicObject Object (Page 1975)
- EllipseSegment Object (Page 1963)
- Ellipse Object (Page 1957)
- Circle Object (Page 1928)
- CheckBox Object (Page 1924)
- Button Object (Page 1921)

**FillingIndex Property**

**Description**

Defines the percentage value (relative to the height of the object) to which to fill an object with closed frame lines (such as a Circle or Rectangle).

The fill level is represented by the current background color. The unfilled background is transparent.
Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the rectangle will be filled to 50%:

Sub RectangleConfiguration()
    'VBA495
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjecit("Rectangle1", "HMIRectangle")
    With objRectangle
        .Filling = True
        .FillingIndex = 50
    End With
End Sub

See also:

PieSegment Object (Page 2032)
FillColor Property (Page 2231)
BackColor Property (Page 2131)
StaticText Object (Page 2067)
Slider object (Page 2062)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
OptionGroup Object (Page 2025)
GraphicObject Object (Page 1975)
EllipseSegment Object (Page 1963)
Ellipse Object (Page 1957)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)

FillingDirection property

Description

0 = the object enclosed in a frame line is filled from bottom to top.
1 = the object enclosed in a frame line is filled from top to bottom.
2 = the object enclosed in a frame line is filled from left to right.
3 = the object enclosed in a frame line is filled from right to left.

Write/Read access.
Use the "FillingDirection" property to set the object fill direction.

Example

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example, the object is filled from left to right.

Sub RectangleConfiguration()
    'VBA906
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .FillingDirection = 2
    End With
End Sub

FillStyle Property

Description

Defines or returns the fill style for the object.

<table>
<thead>
<tr>
<th>Fill pattern</th>
<th>Value</th>
<th>Fill pattern</th>
<th>Value</th>
<th>Fill pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; Transparent &gt;</td>
<td>65536</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; Solid &gt;</td>
<td>0</td>
<td>1048576</td>
<td>196611</td>
<td>196627</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048577</td>
<td>196612</td>
<td>196628</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048578</td>
<td>196613</td>
<td>196629</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048579</td>
<td>196614</td>
<td>196630</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048832</td>
<td>196615</td>
<td>196631</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1048833</td>
<td>196616</td>
<td>196632</td>
<td></td>
</tr>
</tbody>
</table>
### Example

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the fill pattern will be set to the value "196642":

```vba
Sub RectangleConfiguration()
  'VBA496
  Dim objRectangle As HMIRectangle
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
  With objRectangle
    .FillStyle = 196642
  End With
End Sub
```

### See also

- TextList Object (Page 2076)
- StaticText Object (Page 2067)
- Slider object (Page 2062)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
GraphicObject Object (Page 1975)
EllipseSegment Object (Page 1963)
Ellipse Object (Page 1957)
Document Object (Page 1949)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BarGraph Object (Page 1914)

**FillStyle2 Property**

**Description**

Defines or returns the fill pattern of the bar for the BarGraph object.

<table>
<thead>
<tr>
<th>Fill pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>65536</td>
</tr>
<tr>
<td>Solid</td>
<td>0</td>
</tr>
<tr>
<td>1048576</td>
<td>196611</td>
</tr>
<tr>
<td>1048577</td>
<td>196612</td>
</tr>
<tr>
<td>1048578</td>
<td>196613</td>
</tr>
<tr>
<td>1048579</td>
<td>196614</td>
</tr>
<tr>
<td>1048832</td>
<td>196615</td>
</tr>
<tr>
<td>1048833</td>
<td>196616</td>
</tr>
</tbody>
</table>
Example

The "BarGraphConfiguration()" procedure configures In this example the bar pattern will be set to "196642":

Sub BarGraphConfiguration()
 'VBA497
 Dim objBarGraph As HMIBarGraph
 Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
 With objBarGraph
   .FillStyle2 = 196642
 End With
End Sub

See also

BarGraph Object (Page 1914)
FillStyleAlignment property

Description
Defines the alignment of the fill pattern for the process picture.

Normal
The fill pattern refers to the process picture. In runtime, no scaling is performed when opening the picture.

Stretched (window)
The fill pattern refers to the window in the Graphics Designer. In runtime, scaling is performed when opening the picture.

FlashBackColor Property

Description
TRUE, when flashing of the background is activated. BOOLEAN write-read access

Note
A change to the attribute does not automatically deactivate the "Windows Style" attribute.

Example:
The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example, background flashing is activated:

```
Sub RectangleConfiguration()
  'VBA498
  Dim objRectangle As HMIRectangle
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
  With objRectangle
    .FlashBackColor = True
  End With
End Sub
```

See also
RoundButton Object (Page 2052)
StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
FlashBorderColor Property

Description

TRUE, when flashing of the object lines is activated. BOOLEAN write-read access.

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example, flashing of the border is activated:

```vba
Sub RectangleConfiguration()
    'VBA499
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .FlashBorderColor = True
    End With
End Sub
```

See also

StaticText Object (Page 2067)
StatusDisplay Object (Page 2070)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
FlashFlashPicture Property

Description

TRUE, when flashing of the flash picture is activated. BOOLEAN write-read access

Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example, flashing of the Flash Picture is activated:

```
Sub StatusDisplayConfiguration()
'VBA500
Dim objsDisplay As HMIStatusDisplay
Set objsDisplay = ActiveDocument.HMIObjects.AddHMOBject("StatusDisplay1", "HMIStatusDisplay")
With objsDisplay
 .FlashFlashPicture = True
End With
End Sub
```
See also

StatusDisplay Object (Page 2070)

FlashForeColor Property

Description

TRUE, when flashing of the text is activated. BOOLEAN write-read access.

Note

A change to the attribute does not automatically deactivate the "Windows Style" attribute.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example, text flashing is activated:

```vba
Sub ButtonConfiguration()
  'VBA501
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
  With objButton
    .FlashForeColor = True
  End With
End Sub
```

See also

TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
CheckBox Object (Page 1924)
Button Object (Page 1921)

FlashPicReferenced Property

Description

TRUE if the Flash Picture assigned in the Status Display object is to be saved. Otherwise, only the associated object reference is saved. BOOLEAN write-read access.
Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the picture assigned in the Status Display object is to be saved.

Sub StatusDisplayConfiguration()
  'VBA502
  Dim objStatusDisplay As HMIStatusDisplay
  Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("StatusDisplay1", "HMIStatusDisplay")
  With objStatusDisplay
    .FlashPicReferenced = True
  End With
End Sub

See also

StatusDisplay Object (Page 2070)

FlashPicTransColor Property

Description

Defines which color of the bitmap object (.bmp, .dib) assigned to the flash picture should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "FlashPicUseTransColor" property is "True".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the color "Yellow" will be set to "Transparent".

Sub StatusDisplayConfiguration()
'VBA503
Dim objStatusDisplay As HMIStatusDisplay
Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("StatusDisplay1", "HMIStatusDisplay")
With objStatusDisplay
    .FlashPicTransColor = RGB(255, 255, 0)
    .FlashPicUseTransColor = True
End With
End Sub

See also

FlashPicUseTransColor Property (Page 2246)
StatusDisplay Object (Page 2070)

FlashPicture Property

Description

Defines or returns the Flash Picture for the Status Display object.

The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.

The "FlashPicReferenced" property defines in this case whether the flash picture will be saved with the Status Display object or referenced.
Example:

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the picture "Testpicture.BMP" will be used as the flash picture:

```vba
Sub StatusDisplayConfiguration()
'VBA504
Dim objStatusDisplay As HMIStatusDisplay
Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("StatusDisplay1", "HMIStatusDisplay")
With objStatusDisplay
' To use this example copy a Bitmap-Graphic
to the "GraCS"-Folder of the actual project.
'Replace the picturename "Testpicture.BMP" with the name of
'the picture you copied
.FlashPicture = "Testpicture.BMP"
End With
End Sub
```

See also

- FlashPicReferenced Property (Page 2242)
- StatusDisplay Object (Page 2070)

FlashPicture property

**Description**

Specifies which flashing picture is to be displayed for the currently selected status. Pictures with the following formats can be inserted: EMF, WMF, BMP, GIF, JPG.

The flash picture should have the same picture size as the basic picture, otherwise its display is distorted.

FlashPictureState property

**Description**

Only used internally.

See also

- AdvancedStateDisplay object (Page 1905)
**FlashPicUseTransColor Property**

**Description**

TRUE, when the configured color ("FlashPicTransColor" property) of the bitmap objects assigned to the flash picture should be set to "transparent". BOOLEAN write-read access.

**Example:**

The "StatusDisplayConfiguration()" procedure accesses the properties of the Status Display. In this example the color "Yellow" will be set to "Transparent":

```vba
Sub StatusDisplayConfiguration()
  Dim objStatusDisplay As HMIStatusDisplay
  Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("StatusDisplay1", "HMIStatusDisplay")
  With objStatusDisplay
    .FlashPicTransColor = RGB(255, 255, 0)
    .FlashPicUseTransColor = True
  End With
End Sub
```

**See also**

FlashPicTransColor Property (Page 2243)
StatusDisplay Object (Page 2070)

**FlashRate Property**

**Description**

Defines or returns the flash frequency of the "GroupDisplay", "AdvancedAnalogDisplay" and "AdvancedStateDisplay" objects. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>
Note
Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.

Example
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the flash frequency will be set to "Medium":

Sub GroupDisplayConfiguration()
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .FlashRate = 1
  End With
End Sub

See also
GroupDisplay Object (Page 1980)

FlashRateBackColor Property

Description
Defines or returns the flash frequency for the object background. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note
Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.
Example

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the flash frequency for the background will be set to "Medium":

Sub ButtonConfiguration()
'VBA507
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
 .FlashRateBackColor = 1
End With
End Sub

See also

StaticText Object (Page 2067)
Slider object (Page 2062)
TextList Object (Page 2076)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Polygon Object (Page 2035)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
GraphicObject Object (Page 1975)
IOField Object (Page 1992)
EllipseSegment Object (Page 1963)
Ellipse Object (Page 1957)
Circle Object (Page 1928)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BarGraph Object (Page 1914)
**FlashRateBorderColor Property**

**Description**

Defines or returns the flash frequency for the lines of the object. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Note**

Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.

**Example**

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the flash frequency for the border will be set to "Medium":

```vba
Sub ButtonConfiguration()
  'VBA508
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
  With objButton
    .FlashRateBorderColor = 1
  End With
End Sub
```

**See also**

- Slider object (Page 2062)
- StatusDisplay Object (Page 2070)
- StaticText Object (Page 2067)
- TextList Object (Page 2076)
- RoundRectangle Object (Page 2055)
- RoundButton Object (Page 2052)
- Rectangle Object (Page 2049)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
**FlashRateFlashPic Property**

**Description**

Defines or returns the flash frequency for the status display. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.25 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 0.5 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 1 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Note**

Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.
Example

The "GroupDisplayConfiguration()" procedure accesses the properties of the status display.
In this example the flash frequency for the flash picture will be set to "Medium":

```vba
Sub StatusDisplayConfiguration()
    'VBA509
    Dim objStatusDisplay As HMIStatusDisplay
    Set objStatusDisplay = ActiveDocument.HMIObjects.AddHMIObject("StatusDisplay1", "HMIStatusDisplay")
    With objStatusDisplay
        .FlashRateFlashPic = 1
    End With
End Sub
```

See also

StatusDisplay Object (Page 2070)

FlashRateForeColor Property

Description

Defines or returns the flash frequency for the object label. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>Flash frequency</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow (approx. 0.5 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>Medium (approx. 2 Hz)</td>
<td>1</td>
</tr>
<tr>
<td>Fast (approx. 8 Hz)</td>
<td>2</td>
</tr>
</tbody>
</table>

Note

Because the flashing is performed by means of software engineering, the flash frequency is both system-dependent and hardware-bound (number of objects, processor speed, RAM size, update time, etc.).

The information in the table is therefore only for orientation purposes.
Example

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the flash frequency for the label will be set to "Medium":

```vbnet
Sub ButtonConfiguration()
    'VBA510
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
    With objButton
        .FlashRateForeColor = 1
    End With
End Sub
```

See also

TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
CheckBox Object (Page 1924)
Button Object (Page 1921)

FlashState property

Description

Only used internally.

See also

AdvancedAnalogDisplay object (Page 1901)

Folder Property

Description

Returns a folder from the components library.
Example:

The "ShowFolderItems()" procedure accesses the symbol libraries. In this example all the folder names in the global symbol library and project symbol library will be output:

```vba
Sub ShowFolderItems()
'VBA511
Dim colFolderItems As HMIFolderItems
Dim objFolderItem As HMIFolderItem
Dim iAnswer As Integer
Dim iMaxFolder As Integer
Dim iMaxSymbolLib As Integer
Dim iSymbolLibIndex As Integer
Dim iSubFolderIndex As Integer
Dim strSubFolderName As String
Dim strFolderItemName As String

'To determine the number of symbol libraries:
iMaxSymbolLib = Application.SymbolLibraries.Count
iSymbolLibIndex = 1
For iSymbolLibIndex = 1 To iMaxSymbolLib
    With Application.SymbolLibraries(iSymbolLibIndex)
        Set colFolderItems = .FolderItems
        iMaxFolder = .FolderItems.Count
        MsgBox "Number of FolderItems in " & .Name & " : " & iMaxFolder
    End With
Next iSymbolLibIndex

'Output of all subfoldernames from actual folder:
For Each objFolderItem In colFolderItems
    iSubFolderIndex = 1
    For iSubFolderIndex = 1 To iMaxFolder
        strFolderItemName = objFolderItem.DisplayName
        If 0 <> objFolderItem.Folder.Count Then
            strSubFolderName = objFolderItem.Folder(iSubFolderIndex).DisplayName
            iAnswer = MsgBox("SymbolLibrary: " & .Name & vbCrLf & "act. Folder: " & strFolderItemName & vbCrLf & "act. Subfolder: " & strSubFolderName, vbOKCancel)
            If vbCancel = iAnswer Then
                Exit For
            End If
        Else
            MsgBox "There are no subfolders in " & objFolderItem.DisplayName
            Exit For
        End If
    Next iSubFolderIndex
Next objFolderItem
End Sub
```

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA 2253
See also

- SymbolLibraries Object (Listing) (Page 2073)
- SymbolLibrary Object (Page 2074)
- FolderItems Object (Listing) (Page 1973)
- FolderItem Object (Page 1972)
- Accessing the component library with VBA (Page 1646)

FolderItems Property

Description

Returns a listing containing all the folders in the symbol library.
Example:

The "ShowFolderItems()" procedure accesses the symbol libraries. In this example all the folder names in the global symbol library and project symbol library will be output:

Sub ShowFolderItems()
  'VBA512
  Dim colFolderItems As HMIFolderItems
  Dim objFolderItem As HMIFolderItem
  Dim iAnswer As Integer
  Dim iMaxFolder As Integer
  Dim iMaxSymbolLib As Integer
  Dim iSymbolLibIndex As Integer
  Dim iSubFolderIndex As Integer
  Dim strSubFolderName As String
  Dim strFolderItemName As String

  'To determine the number of symbol libraries:
  iMaxSymbolLib = Application.SymbolLibraries.Count
  iSymbolLibIndex = 1
  For iSymbolLibIndex = 1 To iMaxSymbolLib
    With Application.SymbolLibraries(iSymbolLibIndex)
      Set colFolderItems = .FolderItems
      'To determine the number of folders in actual symbol library:
      iMaxFolder = .FolderItems.Count
      MsgBox "Number of FolderItems in " & .Name & " : " & iMaxFolder
      'Output of all subfolder names from actual folder:
      For Each objFolderItem In colFolderItems
        iSubFolderIndex = 1
        For iSubFolderIndex = 1 To iMaxFolder
          strFolderItemName = objFolderItem.DisplayName
          If 0 <> objFolderItem.Folder.Count Then
            strSubFolderName = objFolderItem.Folder(iSubFolderIndex).DisplayName
            iAnswer = MsgBox("SymbolLibrary: " & .Name & vbCrLf & "act. Folder: " & strFolderItemName & vbCrLf & "act. Subfolder: " & strSubFolderName, vbOKCancel)
            'If "Cancel" is clicked, continued with next FolderItem
            If vbCancel = iAnswer Then
              Exit For
            End If
          Else
            MsgBox "There are no subfolders in " & objFolderItem.DisplayName
            Exit For
          End If
        Next iSubFolderIndex
      Next objFolderItem
    End With
  Next iSymbolLibIndex
End Sub
See also

- FolderItem Object (Page 1972)
- SymbolLibraries Object (Listing) (Page 2073)
- SymbolLibrary Object (Page 2074)
- FolderItems Object (Listing) (Page 1973)
- Accessing the component library with VBA (Page 1646)

FontBold Property

Description

TRUE, when the text in the object should be assigned the "bold" attribute. BOOLEAN write-read access.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font attribute will be set to "Bold":

```vba
Sub ButtonConfiguration()
  'VBA513
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjects.AddHMOObject("Button1", "HMIButton")
  With objButton
    .FontBold = True
  End With
End Sub
```

See also

- TextList Object (Page 2076)
- StaticText Object (Page 2067)
- OptionGroup Object (Page 2025)
- IOField Object (Page 1992)
- GroupDisplay Object (Page 1980)
- CheckBox Object (Page 1924)
- Button Object (Page 1921)
- BarGraph Object (Page 1914)
FontItalic Property

Description

TRUE, when the text in the object should be assigned the "italic" attribute. BOOLEAN write-read access.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font attribute will be set to "Italic":

Sub ButtonConfiguration()
    'VBA514
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
    With objButton
        .FontItalic = True
    End With
End Sub

See also

StaticText Object (Page 2067)
TextList Object (Page 2076)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
GroupDisplay Object (Page 1980)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BarGraph Object (Page 1914)

FontName Property

Description

Defines or returns the font name of the text in the object.

All the fonts installed in Windows are available for selection.
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font is set to Arial:

Sub ButtonConfiguration()
'VBA515
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
  .FontName = "Arial"
End With
End Sub

See also

- CheckBox Object (Page 1924)
- TextList Object (Page 2076)
- StaticText Object (Page 2067)
- OptionGroup Object (Page 2025)
- IOField Object (Page 1992)
- GroupDisplay Object (Page 1980)
- Button Object (Page 1921)
- BarGraph Object (Page 1914)

FontSize Property

Description

Defines or returns the font size of the text in the object in points.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font size will be set to 10 points:

Sub ButtonConfiguration()
'VBA516
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
  .FONTSIZE = 10
End With
End Sub
FontUnderline Property

Description

TRUE, when the text in the object should be assigned the "underline" attribute. BOOLEAN write-read access.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font attribute will be set to "Underline":

Sub ButtonConfiguration()
'VBA517
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
  .FontUnderline = True
End With
End Sub
ForeColor Property

Description
Defines or returns the color of the font for the text in the object. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font color will be set to "Red":

```vba
Sub ButtonConfiguration()
'VBA518
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
.ForeColor = RGB(255, 0, 0)
End With
End Sub
```

See also
Button Object (Page 1921)
TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
GroupDisplay Object (Page 1980)
CheckBox Object (Page 1924)
BarGraph Object (Page 1914)
ForeColor_Alarm.._Warning property

Description
Defines the color used for the foreground of one of the following states or message types:
- Alarm
- Warning
- Tolerance
- AS Process Control Error
- AS Control System Fault
- Operator request
- OK
- Simulation

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

ForeFlashColorOff Property

Description
Defines or returns the color of the text for flash status "Off". LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font color when the flash status is "Off" will be set to "Red":

```vba
Sub ButtonConfiguration()
'VBA519
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
  .ForeFlashColorOff = RGB(255, 0, 0)
End With
End Sub
```

See also

CheckBox Object (Page 1924)
TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
GroupDisplay Object (Page 1980)
Button Object (Page 1921)
BarGraph Object (Page 1914)

ForeFlashColorOn Property

Description

Defines or returns the color of the text for flash status "On". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the font color when the flash status is "On" will be set to "White"

Sub ButtonConfiguration()
'VBA520
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
  .ForeFlashColorOn = RGB(255, 255, 255)
End With
End Sub

See also

TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
GroupDisplay Object (Page 1980)
CheckBox Object (Page 1924)
Button Object (Page 1921)
BarGraph Object (Page 1914)

Format property

Description

Specifies the format in which the value is displayed in the advanced analog display.

<table>
<thead>
<tr>
<th>No Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td>Displays a digit or a zero.</td>
</tr>
<tr>
<td>(#)</td>
<td>Displays a digit or no output.</td>
</tr>
<tr>
<td>(.)</td>
<td>Placeholder for decimal character.</td>
</tr>
<tr>
<td>(%)</td>
<td>Placeholder for percentage.</td>
</tr>
<tr>
<td>(,)</td>
<td>Thousand separator.</td>
</tr>
<tr>
<td>((E- E+ e- e+))</td>
<td>Scientific format.</td>
</tr>
<tr>
<td>(- + $ ()</td>
<td>Display of a literal character.</td>
</tr>
<tr>
<td>()</td>
<td>Display the next character in the format character sequence.</td>
</tr>
<tr>
<td>(&quot;ABC&quot;)</td>
<td>Displays the string in inverted commas (&quot; &quot;).</td>
</tr>
</tbody>
</table>
GlobalColorScheme property

**Description**
Defines whether the colors defined for the current design in the global color scheme will be used for this object.

- **yes** Uses the colors from the global color scheme defined for this type of object.
- **No** Uses the colors from the color scheme defined for this type of object under "Colors".

**Example**
```
--
```

GlobalShadow property

**Description**
Defines whether the object will be displayed with the shadowing defined in the active design.

- **yes** Uses the global shadowing defined for this object type.
- **No** No shadowing.

**Example**
```
--
```

GNQBackColorOff..ColorOn property

**Description**
Specifies for the selected message type and the state "Went Out Unacknowledged" which color the background of the value to be displayed assumes for flashing status "Off" (GNQBackColorOff) or "On" (GNQBackColorOn).

**Determination of Color Value**
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).
GNQBackFlash property

Description
Specifies for the selected message type and status "Went Out Unacknowledged" whether the background of the value to be displayed flashes when a message goes out unacknowledged.

GNQTextColorOff..ColorOn property

Description
Specifies for the selected message type and the state "Went Out Unacknowledged" which color the text of the value to be displayed assumes for flashing status "Off" (GNQTextColorOff) or "On" (GNQTextColorOn).

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

GNQTextFlash property

Description
Specifies for the selected message type and status "Went Out Unacknowledged" whether the text of the value to be displayed flashes when a message goes out unacknowledged.

Grid Property

Description
TRUE if the grid is enabled for the active picture. BOOLEAN write-read access.
The grid is only visible during the configuration phase.

Example:
The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the grid for the active picture will be enabled:

```vba
Sub ActiveDocumentConfiguration()
    Application.ActiveDocument.Grid = True
End Sub
```
GridColor Property

Description
Defines or returns the color of the grid during the configuration phase. The Grid property must be set to TRUE for the grid to be displayed. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the grid color for the active picture will be set to "Blue":

Sub ActiveDocumentConfiguration()
  'VBA522
  Application.ActiveDocument.Grid = True
  Application.ActiveDocument.GridColor = RGB(0, 0, 255)
End Sub

See also
Grid Property (Page 2265)
ActiveDocument Property (Page 2107)
Application Property (Page 2121)
GridHeight Property

Description

Defines or returns the height (in pixels) of the grid in the current picture during the configuration phase. The Grid property must be set to TRUE for the grid to be displayed.

Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the grid height for the active picture will be set to "8":

```vba
Sub ActiveDocumentConfiguration()
    'VBA523
    Application.ActiveDocument.Grid = True
    Application.ActiveDocument.GridHeight = 8
End Sub
```

See also

- GridWidth Property (Page 2267)
- Grid Property (Page 2265)
- ActiveDocument Property (Page 2107)
- Application Property (Page 2121)
- Document Object (Page 1949)
- Application Object (Page 1909)

GridWidth Property

Description

Defines or returns the width (in pixels) of the grid in the current picture during the configuration phase. The Grid property must be set to TRUE for the grid to be displayed.
Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the grid width for the active picture will be set to "8":

Sub ActiveDocumentConfiguration()
  'VBA524
  Application.ActiveDocument.Grid = True
  Application.ActiveDocument.GridWidth = 8
End Sub

See also

Grid Property (Page 2265)
GridHeight Property (Page 2267)
ActiveDocument Property (Page 2107)
Application Property (Page 2121)
Document Object (Page 1949)
Application Object (Page 1909)

GroupParent Property

Description

Returns the higher-ranking object in the specified group object. Read-only access.

Example:

--

See also

Group Object (Page 1978)
ActiveDocument Property (Page 2107)
GroupedObjects Object (Listing) (Page 1983)
Document Object (Page 1949)
Application Object (Page 1909)
GroupedHMIObjects Property

Description

Returns a listing containing all the objects in the current group.

Example:

In this example the group object "Group1" is created from a number of objects. An ellipse segment is then added to the group object:

```vba
Sub CreateGroup()
'VBA526
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipseSegment As HMIEllipseSegment
Dim objGroup As HMIGroup
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
End With
With objRectangle
    .Top = 80
    .Left = 80
    .Selected = True
End With
MsgBox "Objects selected!"
Set objGroup = ActiveDocument.Selection.CreateGroup
objGroup.ObjectName = "Group1"
Set objEllipseSegment = ActiveDocument.HMIObjects.AddHMIObject("EllipseSegment", "HMIEllipseSegment")
    'Add one object to the existing group
objGroup.GroupedHMIObjects.Add ("EllipseSegment")
End Sub
```

See also

Group Object (Page 1978)

Height Property

Description

Defines or returns the height of the object (Document, View, Object) in pixels.
Note concerning the Document and View objects:
The default value corresponds to the vertical screen resolution set by the operating system. The specified value can be higher than the current screen resolution. The picture can then be moved with the aid of scroll bars.
The maximum picture height that can be set is 10000 pixels.

Example:
The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the height of the current picture will be set to "1600":

Sub ActiveDocumentConfiguration()
'VBA527
Application.ActiveDocument.Height = 1600
End Sub

See also
View Object (Page 2102)
HMIOBJECT Object (Page 1988)
Document Object (Page 1949)

Hide Property

Description
TRUE if the specified picture is opened as "Visible". BOOLEAN write-read access.
Use the Hide property in order to test, for example, whether a picture is to be visible or invisible when opened. Other WinCC editors (such as CrossReference) open pictures so that they are invisible, i.e. they are not displayed in the Graphics Designer. If you use the DocumentOpened event, for example, you can use the Hide property to prevent the code in the event from being executed by testing that the Hide property is FALSE.
Use the Add and Open methods to define whether a picture is to be visible or invisible when opened.

Note
If you set a picture to "Invisible" (Hide = FALSE), you can then only address it via the Documents listing. The picture is no longer available in the Graphics Designer.
Example:

In the following example, when a picture opens an output indicates whether the picture was opened as visible or invisible:

```vba
Private Sub Document_Opened(CancelForwarding As Boolean)
  'VBA802
  MsgBox Me.Hide
End Sub
```

See also

- Open Method (Page 1867)
- Add Method (Documents Listing) (Page 1790)
- Document Object (Page 1949)

HiddenInput Property

Description

TRUE, when the input value should not be displayed when being entered. Each character entered is substituted by a *. BOOLEAN write-read access.

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the input will be hidden:

```vba
Sub IOFieldConfiguration()
  'VBA528
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .HiddenInput = True
  End With
End Sub
```

See also

- IOField Object (Page 1992)
HMIObjects Property

Description

Returns a listing containing all the objects in the specified picture.

To return an element from the HMIObjects listing you can use either the index number or the object name.

Example:

Use the "AddHMIObject(ObjectName, ProgID)" method to insert a new object in a picture:

```vba
Sub AddCircle()
    'VBA529
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("my Circle", "HMICircle")
End Sub
```

See also

Document Object (Page 1949)

HMIUdoObjects property

Description

Supplies a collection of HMIObject objects that represent the inner objects of the "CustomizedObjects" object.

See also

CustomizedObject Object (Page 1938)

Hotkey Property

Description

Defines or returns the function key for a mouse action in the case of the Button object.

<table>
<thead>
<tr>
<th>Function key</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>112</td>
</tr>
<tr>
<td>F2</td>
<td>113</td>
</tr>
<tr>
<td>F3</td>
<td>114</td>
</tr>
<tr>
<td>Function key</td>
<td>Assigned Value</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>F4</td>
<td>115</td>
</tr>
<tr>
<td>F5</td>
<td>116</td>
</tr>
<tr>
<td>F6</td>
<td>117</td>
</tr>
<tr>
<td>F7</td>
<td>118</td>
</tr>
<tr>
<td>F8</td>
<td>119</td>
</tr>
<tr>
<td>F9</td>
<td>120</td>
</tr>
<tr>
<td>F10</td>
<td>121</td>
</tr>
<tr>
<td>F11</td>
<td>122</td>
</tr>
<tr>
<td>F12</td>
<td>123</td>
</tr>
</tbody>
</table>

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example it is intended that the button can also be launched with function key "F5":

```
Sub ButtonConfiguration()
  'VBA530
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjec.ts.AddHMIObject("Button1", "HMIButton")
  With objButton
    .Hotkey = 116
  End With
End Sub
```

See also

Button Object (Page 1921)

Hysteresis Property

Description

TRUE if the display must include hysteresis (deadband) in the case of the BarGraph object. BOOLEAN write-read access.
Example:

The "BarGraphConfiguration()" procedure configures In this example the display shall take place with hysteresis:

```vba
Sub BarGraphConfiguration()
'VBA531
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObj...<omitted for brevity...>
End With
End Sub
```

See also

BarGraph Object (Page 1914)

HysteresisRange Property

Description

Defines or returns the hysteresis (deadband) as a percentage of the display value.

The Hysteresis property must be set to TRUE for the hysteresis to be calculated.

Example:

The "BarGraphConfiguration()" procedure configures In this example the hysteresis will be set to "4%":

```vba
Sub BarGraphConfiguration()
'VBA532
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObj...<omitted for brevity...>
End With
End Sub
```

See also

BarGraph Object (Page 1914)

Hysteresis Property (Page 2273)
Icon Property

Description

Defines the icon (*.ICO, full path and file name) or returns the path and file name for a button on a user-defined toolbar.

Path specifications

The following path specification formats are possible:

- **Absolute**: z.B. "C:\Siemens\WinCC\Icons\myIcon.ICO."
- **Relative**: The starting folder for relative path specification is the "GraCS" folder of the current project.
- **<global>**: Refers to the installation path for WinCC. The path specification "<global>\Icons\myIcon" is the same as the path specification under "Absolute".
- **<project>**: Refers to the current project directory (see example).

Example:

The "CreateToolbar()" procedure creates a user-defined toolbar with two icons:

```vba
Sub CreateToolbar()
    'VBA533
    Dim objToolbar As HMIToolbar
    Dim objToolbarItem As HMIToolbarItem
    Dim strFileWithPath
    Set objToolbar = ActiveDocument.CustomToolbars.Add("Tool1_1")
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "ti1_1", "myFirstToolbaritem")
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(2, "ti1_2", "mySecondToolbaritem")
    'To use this example copy a *.ICO-Graphic to the "GraCS"-Folder of the actual project.
    'Replace the filename "EZSTART.ICO" in the next commandline
    'with the name of the ICO-Graphic you copied
    strFileWithPath = Application.ApplicationDataPath & "EZSTART.ICO"
    'To assign the symbol-icon to the first toolbaritem
    objToolbar.ToolbarItems(1).Icon = strFileWithPath
End Sub
```
See also

ToolbarItems Object (Listing) (Page 2085)
ToolbarItem Object (Page 2083)
How to Add a New Icon to the Toolbar (Page 1637)
How to Create an Application-specific Toolbar (Page 1635)

IndependentWindow property

Description

Defines whether the display of the picture window in Runtime depends on the process picture in which the picture window was configured.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>Size and position of the picture window are independent of the process picture and only defined by the &quot;Window mode&quot; attribute</td>
</tr>
<tr>
<td>No</td>
<td>Size and position of the picture window change with the shift or scaling of the process picture</td>
</tr>
</tbody>
</table>

Index Property

Description

Status display
Defines the status (0 bis 255) or returns it. A basic picture and flash picture can be defined for each status value.

Line Object
Defines the start and end point for a line, and so also defines the direction. Use the ActualPointLeft and ActualPointTop properties to define the coordinates for each starting and finishing point.

Polygon object, PolyLine object and TubePolyline object
Defines or returns the number of the corner point whose position coordinates you want to change or display.

CheckBox and OptionGroup objects
Defines or returns the number (1 to 32) of the field whose text is to be defined.

ComboBox and ListBox object
Defines or returns the number (1 to 32) of the line whose text is to be defined.
Example 1: Line

In the following example a line will be inserted into the active picture and the starting and finishing points will be defined:

Sub LineAdd()
'VBA682
Dim objLine As HMILine
Dim objEvent As HMIEvent
Set objLine = ActiveDocument.HMIObjects.AddHMIObject("myLine", "HMILine")
With objLine
    .BorderColor = RGB(255, 0, 0)
    .Index = hmiLineIndexTypeStartPoint
    .ActualPointLeft = 12
    .ActualPointTop = 34
    .Index = hmiLineIndexTypeEndPoint
    .ActualPointLeft = 74
    .ActualPointTop = 64
End With
End Sub

Example 2: Polyline

For this example to work, insert a polyline called "Polyline1" into the active picture: The "PolyLineCoordsOutput" procedure then outputs the coordinates of all the corner points in the polyline:

Sub PolyLineCoordsOutput()
'VBA534
Dim iPcIndex As Integer
Dim iPosX As Integer
Dim iPosY As Integer
Dim iIndex As Integer
Dim objPolyLine As HMIPolyLine
Set objPolyLine = Application.ActiveDocument.HMIObjects.AddHMIObject("Polyline1", "HMIPolyLine")

'Determine number of corners from "Polyline1":
iPcIndex = objPolyLine.PointCount

'Output of x/y-coordinates from every corner:
For iIndex = 1 To iPcIndex
    With objPolyLine
        .Index = iIndex
        iPosX = .ActualPointLeft
        iPosY = .ActualPointTop
        MsgBox iIndex & ". corner:" & vbCrLf & "x-coordinate: " & iPosX & vbCrLf & "y-coordinate: " & iPosY
    End With
Next iIndex
End Sub
Example 3: Check box

The "CreateOptionGroup()" procedure creates the OptionGroup object with four option buttons. Each option button is assigned the default name "myCustomText<Nummer>":

```vba
Sub CreateOptionGroup()
'VBA535
Dim objRadioBox As HMIOptionGroup
Dim iIndex As Integer
Set objRadioBox = ActiveDocument.HMIObjects.AddHMIObject("RadioBox_1", "HMIOptionGroup")
With objRadioBox
  .Height = 100
  .Width = 180
  .BoxCount = 4
  For iIndex = 1 To .BoxCount
    .index = iIndex
    .Text = "myCustomText" & .index
  Next iIndex
End With
End Sub
```

See also

- Line Object (Page 2004)
- FlashPicture Property (Page 2244)
- BasePicture Property (Page 2144)
- ActualPointTop Property (Page 2110)
- ActualPointLeft Property (Page 2109)
- StatusDisplay Object (Page 2070)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- OptionGroup Object (Page 2025)

InheritState property

Description

Defines whether the "Display" and "Operator Control Enable" properties of the user object can be inherited by the individual objects of the user object.
InputValue property

Description

Defines the value to be entered by the user in the I/O field. The value is not displayed in the I/O field when the property is set.

If you want the value to be displayed in the I/O field after confirmation with the <Return> key, configure a direct connection between the properties "input value" and "output value". The direct connection is only practical when no tag is connected to the output value, but the user can nevertheless query the specified value, for example, through a script.

Example:

IsActive Property

Description

Returns TRUE if a copy of the current picture is active. BOOLEAN read access.

Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example a copy of the current picture will be created and an output will indicate whether the copy is active.

Sub ActiveDocumentConfiguration()
  'VBA537
  Application.ActiveDocument.Views.Add
  'If you comment out the following line
  'and recall the procedure, the output of
  'the messagebox is different
  'Output state of copy:
  MsgBox Application.ActiveDocument.Views(1).IsActive
End Sub
See also

ActiveDocument Property (Page 2107)
View Object (Page 2102)

IsConnectedToProject Property

Description

Returns TRUE if the project connection is available. BOOLEAN read access.

Example:

The "ConnectCheck()" procedure checks whether a project connection exists and outputs the result:

```vba
Sub ConnectCheck()
'VBA538
Dim bCheck As Boolean
Dim strStatus As String
bCheck = Application.IsConnectedToProject
If bCheck = True Then
    strStatus = "yes"
Else
    strStatus = "no"
End If
MsgBox "Connection to project available: " & strStatus
End Sub
```

See also

Application Object (Page 1909)

IsDynamicable Property

Description

TRUE if a property can be made dynamic. BOOLEAN read access.
Example:

The HMIObjectPropertyChanged event always occurs when you change an object property in the Graphics Designer. In this example the property name and value will be output. A check will also be made on whether the property can be made dynamic:

Sub Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As Boolean)
  'VBA539
  Dim objProp As IHMIProperty
  Dim strStatus As String
  Set objProp = Property
  'Checks whether property is dynamicable
  If objProp.IsDynamicable = True Then
    strStatus = "yes"
  Else
    strStatus = "no"
  End If
  MsgBox "Property: " & objProp.Name & vbCrLf & "Value: " & objProp.value & vbCrLf & "Dynamicable: " & strStatus
End Sub

Further information on the "Events" topic can be found under the heading "Executing VBA macros in Graphics Designer".

See also

Property Object (Page 2042)
HMIOBJECT Object (Page 1988)
HMIOBJECTPROPERTYCHANGED Event (Page 1770)
Executing VBA Macros in Graphics Designer (Page 1619)

IsPublished property

Description

Only used internally.

See also

FaceplateProperty object (Page 1971)
Property Object (Page 2042)
Italic Property

Description

TRUE if the font attribute "Italic" is set for the language-dependent text in the object. BOOLEAN write-read access.

Example:

The following example sets the font attributes of a button for French and English:

Sub ExampleForLanguageFonts()
    'VBA540
    Dim objLangFonts As HMILanguageFonts
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
    objButton.Text = "Hello"
    Set objLangFonts = objButton.LDFonts
    'To make fontsettings for french:
    With objLangFonts.ItemByLCID(1036)
        .Family = "Courier New"
        .Bold = True
        .Italic = False
        .Underlined = True
        .Size = 12
    End With
    'To make fontsettings for english:
    With objLangFonts.ItemByLCID(1033)
        .Family = "Times New Roman"
        .Bold = False
        .Italic = True
        .Underlined = False
        .Size = 14
    End With
End Sub

See also

Underlined Property (Page 2460)
Size Property (Page 2419)
Parent Property (Page 2359)
LanguageID Property (Page 2289)
Family Property (Page 2229)
Bold Property (Page 2151)
Item Property

Description

Returns an element from a listing. Depending on the specified object, you can use either the index number or the name to return a particular element.

Example:

This example shows both kinds of indexing. In order for the example to work, create a group object ("Group1") with two objects. The example outputs the height of the second object in a group:

```vba
Sub GetHeight()
  'VBA541
  Dim objGroup As HMIGroup
  'Next line uses the property "Item" to get a group by name
  Set objGroup = ActiveDocument.HMIObjects.Item("Group1")
  'Otherwise next line uses index to identify a groupobject
  MsgBox "The height of object 2 is: " & objGroup.GroupedHMIObjects.Item(2).Height
End Sub
```

See also

- VariableTriggers Object (Listing) (Page 2101)
- VariableStateValues Object (Listing) (Page 2097)
- AnalogResultInfos Object (Listing) (Page 1908)

ItemBorderBackColor Property

Description

Defines or returns the background color of the separation lines in the selection list for the TextList object. LONG write-read access.

The background color is only visible with the property setting ItemBorderStyle > 0.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the background color for the separation lines will be set to "Red":

```
Sub TextListConfiguration()
    'VBA542
    Dim objTextList As HMITextList
    Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
    With objTextList
        .ItemBorderStyle = 1
        .ItemBorderBackColor = RGB(255, 0, 0)
    End With
End Sub
```

See also

- ItemBorderStyle Property (Page 2285)
- TextList Object (Page 2076)

**ItemBorderColor Property**

**Description**

Defines or returns the color of the separation lines in the selection list for the TextList object. LONG write-read access.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the color for the separation lines will be set to "White":

Sub TextListConfiguration()
  'VBA543
  Dim objTextList As HMITextList
  Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
  With objTextList
    .ItemBorderStyle = 1
    .ItemBorderColor = RGB(255, 255, 255)
  End With
End Sub

See also

TextList Object (Page 2076)

ItemBorderStyle Property

Description

Defines or returns the dividing line style in the selection list for the TextList object. Value range from 0 to 4.

<table>
<thead>
<tr>
<th>Line style</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>------</td>
<td>0</td>
</tr>
<tr>
<td>---</td>
<td>1</td>
</tr>
<tr>
<td>-------</td>
<td>2</td>
</tr>
<tr>
<td>--</td>
<td>3</td>
</tr>
<tr>
<td>------</td>
<td>4</td>
</tr>
</tbody>
</table>
Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the dividing line style will be set to "1":

```vba
Sub TextListConfiguration()
  'VBA544
  Dim objTextList As HMITextList
  Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
  With objTextList
    .ItemBorderStyle = 1
    .ItemBorderBackColor = RGB(255, 0, 0)
  End With
End Sub
```

See also

TextList Object (Page 2076)

ItemBorderWidth Property

Description

Defines or returns the weight in pixels of the dividing lines in the selection list for the TextList object.

Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the dividing line width will be set to "4":

```vba
Sub E_628_TextListConfiguration()
Sub E_629_TextListConfiguration()
  'VBA545
  Dim objTextList As HMITextList
  Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
  With objTextList
    .ItemBorderWidth = 4
  End With
End Sub
```
See also

TextList Object (Page 2076)

Key Property

Description

Returns the name that identifies the entry (menu point or icon) in the user-defined menu or user-defined toolbar. Read only access.

Use the Key property to determine which entry was clicked. For this purpose you can use, say, the events "MenuItemClicked" and "ToolBarItemClicked".

Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"):

```vba
Sub CreateMenuItem()
'VBA546
Dim objMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
',
'Add new menu "Delete objects" to menubar:
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")
',
'Adds two menuitems to menu "Delete objects"
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete Rectangles")
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete Circles")
End Sub
```
In connection with the "MenuItemClicked" event, you can connect the menu entries with procedure calls, for instance. In this example the names of the menu entries will be output:

```vba
Sub Document_MenuItemClicked(ByVal MenuItem As IHMMenuItem)
    'VBA547
    Dim strClicked As String
    Dim objMenuItem As HMIMenuItem
    Set objMenuItem = MenuItem
    'strClicked can get two values:
    '(1) "DeleteAllRectangles" and
    '(2) "DeleteAllCircles"
    strClicked = objMenuItem.Key
    'To analyse "strClicked" with "Select Case"
    Select Case strClicked
        Case "DeleteAllRectangles"
            'Instead of "MsgBox" a procedurecall (e.g. "Call <Prozedurname>") can stay here
            MsgBox "'Delete rectangle' was clicked"
        Case "DeleteAllCircles"
            MsgBox "'Delete Circles' was clicked"
    End Select
End Sub
```

See also

- ToolbarItem Object (Page 2083)
- MenuItem Object (Page 2013)
- InsertToolbarItem Method (Page 1856)
- InsertMenuItem Method (Page 1852)
- ToolbarItemClicked Event (Page 1782)
- MenuItemClicked Event (Page 1775)
- Creating Customized Menus and Toolbars (Page 1627)

Label Property

Description

Returns the label of the user-defined menu or menu entry in the currently set language. Read only access.
Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"): In this example the labels will then be output:

Sub CreateMenuItem()
'VBA548
Dim objMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Dim iIndex As Integer
iIndex = 1

'Add new menu "Delete objects" to menubar
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")

'Adds two menuitems to menu "Delete objects"
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete rectangles")
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete circles")
MsgBox ActiveDocument.CustomMenus(1).Label
For iIndex = 1 To objMenu.MenuItems.Count
MsgBox objMenu.MenuItems(iIndex).Label
Next iIndex
End Sub

See also

- CustomMenus Property (Page 2207)
- MenuItems Object (Listing) (Page 2015)
- MenuItem Object (Page 2013)
- Menu Object (Page 2009)

LanguageID Property

Description

Returns the language identifier of the project language as a decimal value. LONG read access
Example:

The "DataLanguages()" procedure outputs the project languages together with their language identifiers:

```
Sub DataLanguages()
    'VBA549
    Dim colDataLang As HMIDataLanguages
    Dim objDataLang As HMIDataLanguage
    Dim nLangID As Long
    Dim strLangName As String
    Dim iAnswer As Integer
    Set colDataLang = Application.AvailableDataLanguages
    For Each objDataLang In colDataLang
        nLangID = objDataLang.LanguageID
        strLangName = objDataLang.LanguageName
        iAnswer = MsgBox(nLangID & " " & strLangName, vbOKCancel)
        If vbCancel = iAnswer Then Exit For
    Next objDataLang
End Sub
```

See also

DataLanguages Object (Listing) (Page 1942)
DataLanguage Object (Page 1941)

LanguageName Property

Description

Returns the project language. STRING read access.
Example:

The "DataLanguages()" procedure outputs the project languages together with their language identifiers:

```vba
Sub DataLanguages()
'VBA550
Dim colDataLang As HMIDataLanguages
Dim objDataLang As HMIDataLanguage
Dim nLangID As Long
Dim strLangName As String
Dim iAnswer As Integer
Set colDataLang = Application.AvailableDataLanguages
For Each objDataLang In colDataLang
    nLangID = objDataLang.LanguageID
    strLangName = objDataLang.LanguageName
    iAnswer = MsgBox(nLangID & " ", vbOKCancel)
    If vbCancel = iAnswer Then Exit For
Next objDataLang
End Sub
```

See also

DataLanguages Object (Listing) (Page 1942)
DataLanguage Object (Page 1941)

LanguageSwitch Property

Description

Defines where the language-dependent assignment texts are stored or returns the value.
BOOLEAN write-read access.
TRUE, when the texts in the Text Library are managed. Translation to other language occurs in the Text Library.
FALSE, when the texts are managed directly in the object. Translation to other language can be carried out using Text Distributor.
Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the texts will be managed in the Text Library:

```vba
Sub TextListConfiguration()
'VBA551
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
  .LanguageSwitch = True
End With
End Sub
```

See also

TextList Object (Page 2076)

LastChange Property

Description

Returns the date on which the current picture was last changed. READ access.

Example:

The "ActiveDocumentConfiguration()" procedure accesses the properties of the current picture in the Graphics Designer. In this example the date of the last change to the current picture will be output:

```vba
Sub ActiveDocumentConfiguration()
'VBA552
Dim varLastDocChange As Variant
varLastDocChange = Application.ActiveDocument.LastChange
MsgBox "Last changing: " & varLastDocChange
End Sub
```
Layer Property

Description
Defines which layer of the picture an object is located in, or returns that information. There is a total of 32 layers available, whereby Layer "0" is the bottom layer and Layer "31" the top layer.

The configured objects are initially in the background of a layer.

Note
In VBA the numbering starts at "1". An entry of "objRectangle.Layer = 1" is therefore located in the lowest layer.

Example:
The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the Rectangle object will be inserted in layer "4":

Sub RectangleConfiguration()
'VBA553
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
 .Layer = 4
End With
End Sub

See also
HMIOBJECT OBJECT (Page 1988)
Editing Layers with VBA (Page 1657)

Layer00..10Checked property

Description
TRUE if the respective limit "0" to "10" is monitored in the case of the "3DBarGraph" object. BOOLEAN write-read access.
The limit and the color representation are specified with the properties "Layer00..10Value" and "Layer00..10Color".

The bar fill color and the fill pattern are specified with the properties "Layer00..10FillColor" and "Layer00..10FillStyle".

**Example**

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit "0" is to be monitored:

```vba
Sub HMI3DBarGraphConfiguration()
  'VBA554
  Dim obj3DBar As HMI3DBarGraph
  Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
  With obj3DBar
    .Layer00Checked = True
  End With
End Sub
```

**See also**

Layer00..10Value property (Page 2296)
Layer00..10Color property (Page 2294)
3DBarGraph Object (Page 1894)

**Layer00..10Color property**

**Description**

Defines or returns the color for the respective limit "0" to "10" of the "3DBarGraph" object. LONG write-read access.

When monitoring of the limit value is activated using the "Layer00..10Checked" property, the bar turns to the color defined by this attribute on reaching the limit value.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).
Example

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the color for limit "0" is defined as "Magenta":

```vba
Sub HMI3DBarGraphConfiguration()
    'VBA555
    Dim obj3DBar As HMI3DBarGraph
    Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
    With obj3DBar
        .Layer00Checked = True
        .Layer00Color = RGB(255, 0, 255)
    End With
End Sub
```

See also

Layer00..10Checked property (Page 2293)

3DBarGraph Object (Page 1894)

Layer00..10FillColor property

Description

Defines or returns the bar fill color for the respective limit "0" to "10" of the "3DBarGraph" object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0).

Layer00..10FillStyle property

Description

Defines or returns the bar fill pattern for the respective limit "0" to "10" of the "3DBarGraph" object.

The bar fill color has to differ from the bar color to make the fill pattern visible.

There is a choice of 50 fill patterns. The "0" fill pattern fills the object with the set background color. The "1" fill pattern means neither a background nor a fill pattern is displayed.
Layer00..10Value property

Description
Defines or returns the value for "Limit 0" to "Limit 10" in the case of the "3DBarGraph" object. Monitoring only takes effect when the "Layer00..10Checked" property value is set to "TRUE".

Example
The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit "0" is defined as "0":

```vba
Sub HMI3DBarGraphConfiguration()
'VBA556
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
    .Layer00Checked = True
    .Layer00Value = 0
End With
End Sub
```

See also
Layer00..10Checked property (Page 2293)
3DBarGraph Object (Page 1894)

LayerDecluttering Property

Description
TRUE if showing and hiding objects dependent upon the minimum and maximum zoom set for a layer has been enabled. BOOLEAN write-read access.
Example:

In the following example the settings for the lowest layer are configured in the active picture:

```
Sub ConfigureSettingsOfLayer()
  'VBA587
  Dim objLayer As HMILayer
  Set objLayer = ActiveDocument.Layers(1)
  With objLayer
    'configure "Layer 0"
    .MinZoom = 10
    .MaxZoom = 100
    .Name = "Configured with VBA"
  End With
  'define fade-in and fade-out of objects:
  With ActiveDocument
    .LayerDecluttering = True
    .ObjectSizeDecluttering = True
    .SetDeclutterObjectSize 50, 100
  End With
End Sub
```

See also

- Document Object (Page 1949)
- Editing Layers with VBA (Page 1657)

Layers Property

Description

Returns a listing containing the properties of the layers in the current picture.

Note

If the "Layers" property is used, the sequence of HMI objects in the HMIObjecst listing can change.
Example:

The "LayerInfo()" procedure outputs the name and zoom configuration for each layer of the current picture:

Sub LayerInfo()
    'VBA588
    Dim colLayers As HMILayers
    Dim objLayer As HMILayer
    Dim iAnswer As Integer
    Set colLayers = ActiveDocument.Layers
    For Each objLayer In colLayers
        With objLayer
            iAnswer = MsgBox("Layername: " & .Name & vbCrLf & "max. zoom:  " & .MaxZoom & vbCrLf & "min. zoom:  " & .MinZoom, vbOKCancel)
        End With
        If vbCancel = iAnswer Then Exit For
    Next objLayer
    End Sub

See also

Name Property (Page 2343)
MinZoom Property (Page 2340)
MaxZoom Property (Page 2323)
Layers Object (Listing) (Page 2003)
Layer Object (Page 2001)

LDAssignments property

Description

Returns a listing with the (foreign language) assignments of display texts that are displayed depending on the current "Output Value" in the "TextList" object.

The assignments depend on the set list type. Specify the list type with the "ListType" property.

LDFonts Property

Description

Returns a listing containing the language identifiers for the configured fonts.
Example:

Use the LDFonts property to return the LanguageFonts listing. In the following example the
language identifiers of the configured fonts will be output:

Sub ShowLanguageFont()
  'VBA589
  Dim colLanguageFonts As HMI LanguageFonts
  Dim objLanguageFont As HMI LanguageFont
  Dim objButton As HMI Button
  Dim iMax As Integer
  Dim iAnswer As Integer
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
  Set colLanguageFonts = objButton.LDFonts
  iMax = colLanguageFonts.Count
  For Each objLanguageFont In colLanguageFonts
    iAnswer = MsgBox("Projected fonts: " & iMax & vbCrLf & "Language-ID: " &
                     objLanguageFont.LanguageID, vbOKCancel)
    If vbCancel = iAnswer Then Exit For
  Next objLanguageFont
End Sub

See also

StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
LanguageFonts Object (Listing) (Page 1997)
CheckBox Object (Page 1924)
Button Object (Page 1921)

LDFontsType property

Description

Only used internally.

See also

FaceplateProperty object (Page 1971)

LDLabelTexts Property

Description

Returns a listing containing the multilingual labels of the user-defined menu or menu entry.
Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"): In this example, multilingual menu labels will be created:

```vba
Sub CreateMenuItem()
'VBA590
Dim objMenu As HMIMenu
Dim objMenuItems As HMIMenuItem
Dim objLangText As HMILanguageText

'Add new menu "Delete objects" to menubar:
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")

'Add two menuitems to the new menu
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete rectangles")
Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete circles")

'Define foreign-language labels for menu "Delete objects":
Set objLangText = objMenu.LDLabelTexts.Add(1033, "English_Delete objects")
Set objLangText = objMenu.LDLabelTexts.Add(1032, "Greek_Delete objects")
Set objLangText = objMenu.LDLabelTexts.Add(1034, "Spanish_Delete objects")
Set objLangText = objMenu.LDLabelTexts.Add(1036, "French_Delete objects")
End Sub
```

The "LDLabelInfo()" procedure outputs the labels configured for the "Delete Objects" menu:

```vba
Sub LDLabelInfo()
'VBA591
Dim colLangTexts As HMILanguageTexts
Dim objLangText As HMILanguageText
Dim iAnswer As Integer

'Save all labels of menu into collection "colLangTexts":
Set colLangTexts = ActiveDocument.CustomMenus("DeleteObjects").LDLabelTexts
For Each objLangText In colLangTexts
  iAnswer = MsgBox(objLangText.DisplayName, vbOKCancel)
  If vbCancel = iAnswer Then Exit For
Next objLangText
End Sub
```

See also

MenuItem Object (Page 2013)
Menu Object (Page 2009)
LDNames Property

Description

Returns a listing containing the multilingual names of a folder in the Components Library or of a layer.

Example:

Use the LDNames property to return the LanguageTexts listing. In the following example all multilingual layer names will be output:

Explanation: What the example shows

Sub LDLabelInfo()
    'VBA592
    Dim colLayerLngTexts As HMILanguageTexts
    Dim objLayerLngText As HMILanguageText
    Dim iIndex As Integer
    Dim iAnswer As Integer
    Dim strResult As String
    iIndex = 1
    For iIndex = 1 To ActiveDocument.Layers.Count
        'Save all labels of layers into collection of "colLayerLngTexts":
        Set colLayerLngTexts = ActiveDocument.Layers(iIndex).LDNames
        For Each objLayerLngText In colLayerLngTexts
            strResult = strResult & vbCrLf & objLayerLngText.LanguageID & " - " &
            objLayerLngText.DisplayName
        Next objLayerLngText
        iAnswer = MsgBox(strResult, vbOKCancel)
        strResult = ""
        If vbCancel = iAnswer Then Exit For
    Next iIndex
End Sub

See also

Layer Object (Page 2001)
LanguageTexts Object (Listing) (Page 2000)
FolderItem Object (Page 1972)

LDStatusTexts Property

Description

Returns a listing containing the multilingual status line texts of a user-defined icon or menu entry.
Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"). In this example, multilingual status line texts will be created:

Sub CreateMenuItem()
'VBA593
Dim objMenu As HMIMenu
Dim objMenuItem1 As HMIMenuItem
Dim objMenuItem2 As HMIMenuItem
Dim objLangStateText As HMIListItemText

'Add new menu "Delete objects" to menubar:
Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")

'Add two menutems to the new menu
Set objMenuItem1 = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete rectangles")
Set objMenuItem2 = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete circles")

'Define foreign-language labels for menuitem "Delete rectangles":
Set objLangStateText = objMenuItem1.LDStatusTexts.Add(1033, "English_Delete rectangles")
Set objLangStateText = objMenuItem1.LDStatusTexts.Add(1032, "Greek_Delete rectangles")
Set objLangStateText = objMenuItem1.LDStatusTexts.Add(1034, "Spanish_Delete rectangles")
Set objLangStateText = objMenuItem1.LDStatusTexts.Add(1036, "French_Delete rectangles")
End Sub

The "LDStatusTextInfo()" procedure outputs the status line texts configured for the "Delete Objects" menu:

Sub LDStatusTextInfo()
'VBA594
Dim colMenuItems As HMIMenuItems
Dim objMenuItem As HMIMenuItem
Dim colStatusLngTexts As HMIListItemTexts
Dim objStatusLngText As HMIListItemText
Dim strResult As String
Dim iAnswer As Integer
Set colMenuItems = ActiveDocument.CustomMenus("DeleteObjects").MenuItems
For Each objMenuItem In colMenuItems
strResult = "Statustexts of menuitem " & objMenuItem.Label & ""
Set colStatusLngTexts = objMenuItem.LDStatusTexts
For Each objStatusLngText In colStatusLngTexts
strResult = strResult & vbCrLf & objStatusLngText.DisplayName
Next objStatusLngText
iAnswer = MsgBox(strResult, vbOKCancel)
If vbCancel = iAnswer Then Exit For
Next objMenuItem
End Sub
See also

ToolbarItem Object (Page 2083)
MenuItem Object (Page 2013)
Menu Object (Page 2009)

LDTexts Property

Description

Returns a listing containing the multilingual labels of an object.

Example:

The "LDTextInfo()" procedure outputs the labels configured for the Button object. For this example to work, create the object "myButton" in the Graphics Designer and configure a number of multilingual labels:

```vba
Sub LDTextInfo()
  'VBA595
  Dim colLDLngTexts As HMILanguageTexts
  Dim objLDLngText As HMILanguageText
  Dim objButton As HMIButton
  Dim iAnswer As Integer
  Set objButton = ActiveDocument.HMIObjects("myButton")
  Set colLDLngTexts = objButton.LDTexts
  For Each objLDLngText In colLDLngTexts
    iAnswer = MsgBox(objLDLngText.DisplayName, vbOKCancel)
    If vbCancel = iAnswer Then Exit For
  Next objLDLngText
End Sub
```

See also

Button Object (Page 1921)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
CheckBox Object (Page 1924)

LDTooltipTexts Property

Description

Returns a listing containing the multilingual Tooltip texts for a user-defined icon or for an object.
Example

The "CreateToolbar()" procedure creates a user-defined toolbar with two icons. Two multilingual Tooltip texts are assigned to the first icon:

```vba
Sub CreateToolbar()
'VBA596
Dim objToolbar As HMIToolbar
Dim objToolbarItem As HMIToolbarItem
Dim objLangText As HMILanguageText
Dim strFileWithPath
'Create toolbar with two toolbar-items:
Set objToolbar = ActiveDocument.CustomToolbars.Add("Tool1_1")
Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "ti1_1", "myFirstToolbaritem")
Set objToolbarItem = objToolbar ToolbarItems.InsertToolbarItem(2, "ti1_2", "mySecondToolbaritem")
'In order that the example runs correct copy a *.ICO-Graphic into the "GraCS"-Folder of the actual project.
'Replace the filename "EZSTART.ICO" in the next commandline with the name of the ICO-Graphic you copied
strFileWithPath = Application.ApplicationDataPath & "EZSTART.ICO"
'To assign the symbol-icon to the first toolbaritem
objToolbar.ToolbarItems(1).Icon = strFileWithPath
'Define foreign-language tooltiptexts
Set objLangText = objToolbar.ToolbarItems(1).LDTooltipTexts.Add(1036, "French_Tooltiptext")
Set objLangText = objToolbar.ToolbarItems(1).LDTooltipTexts.Add(1034, "Spanish_Tooltiptext")
End Sub
```

The "LDTooltipInfo()" procedure outputs all the Tooltip texts configured for the first icon in the first user-defined toolbar:

```vba
Sub LDTooltipInfo()
'VBA597
Dim colLangTexts As HMILanguageTexts
Dim objLangText As HMILanguageText
Dim iAnswer As Integer
Set colLangTexts = ActiveDocument.CustomToolbars(1).ToolbarItems(1).LDTooltipTexts
For Each objLangText In colLangTexts
iAnswer = MsgBox(objLangText.DisplayName, vbOKCancel)
If vbCancel = iAnswer Then Exit For
Next objLangText
End Sub
```
See also

ToolbarItem Object (Page 2083)
HMIOBJECT Object (Page 1988)

Left Property

Description

Defines or returns the X coordinate of the object (measured from the top left-hand edge of the picture) in pixels. The X-coordinate relates to the top left corner of the rectangle enclosing the object.

View Object

Defines or returns the X coordinate of the window (measured from the top left-hand edge of the Graphics Designer working area) in pixels.

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the rectangle will be moved 40 pixels to the right:

Sub RectangleConfiguration()
'VBA598
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
 .Left = 40
End With
End Sub

See also

View Object (Page 2102)
HMIOBJECT Object (Page 1988)

LeftComma Property

Description

Defines or returns the number of digits to the left of the decimal point (0 to 20) for the BarGraph object.
Example:
The "BarGraphConfiguration()" procedure configures In this example the number of digits to the left of the decimal point will be set to "4".

Sub BarGraphConfiguration()
'VBA599
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  .LeftComma = 4
End With
End Sub

See also
BarGraph Object (Page 1914)

LightEffect Property

Description
TRUE if the light effect of the 3DBarGraph object is activated. BOOLEAN write-read access.

Example:
The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the light effect will be activated:

Sub HMI3DBarGraphConfiguration()
'VBA600
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
  .LightEffect = True
End With
End Sub

See also
3DBarGraph Object (Page 1894)
LimitHigh4 Property

Description

Defines or returns the high limit value for "Reserve 4" in the case of the BarGraph object.

The CheckLimitHigh4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored.

The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh4 property.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "70".

```vba
Sub BarGraphLimitConfiguration()
    'VBA601
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        '.TypeLimitHigh4 = False
        '.CheckLimitHigh4 = True
        '.ColorLimitHigh4 = RGB(255, 0, 0)
        '.LimitHigh4 = 70
    End With
End Sub
```

See also

TypeLimitHigh4 Property (Page 2453)
CheckLimitHigh4 Property (Page 2172)
BarGraph Object (Page 1914)

LimitHigh5 Property

Description

Defines or returns the high limit value for "Reserve 5" in the case of the BarGraph object.

The CheckLimitHigh5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored.

The type of the evaluation (in percent or absolute) is defined in the TypeLimitHigh5 property.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "80".

Sub BarGraphLimitConfiguration()
'VBA602
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  .TypeLimitHigh5 = False
  .CheckLimitHigh5 = True
  .ColorLimitHigh5 = RGB(0, 0, 0)
  .LimitHigh4 = 80
End With
End Sub

See also

TypeLimitHigh5 Property (Page 2453)
CheckLimitHigh5 Property (Page 2173)
BarGraph Object (Page 1914)

LimitLow4 Property

Description

Defines or returns the low limit value for "Reserve 4" in the case of the BarGraph object.
The CheckLimitLow4 property must be set to TRUE in order that the "Reserve 4" limit value can be monitored.
The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow4 property.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "5".

Sub BarGraphLimitConfiguration()
  'VBA603
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis to absolute
    .TypeLimitLow4 = False
    'Activate monitoring
    .CheckLimitLow4 = True
    'Set barcolor to "green"
    .ColorLimitLow4 = RGB(0, 255, 0)
    'Set lower limit to "5"
    .LimitLow4 = 5
  End With
End Sub

See also

CheckLimitLow4 Property (Page 2174)
TypeLimitLow4 Property (Page 2454)
BarGraph Object (Page 1914)

LimitLow5 Property

Description

Defines or returns the low limit value for "Reserve 5" in the case of the BarGraph object.

The CheckLimitLow5 property must be set to TRUE in order that the "Reserve 5" limit value can be monitored.

The type of the evaluation (in percent or absolute) is defined in the TypeLimitLow5 property.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "0".

Sub BarGraphLimitConfiguration()
'VBA604
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis to absolute
  .TypeLimitLow5 = False
  'Activate monitoring
  .CheckLimitLow5 = True
  'Set barcolor to "white"
  .ColorLimitLow5 = RGB(255, 255, 255)
  'Set lower limit to "0"
  .LimitLow5 = 0
End With
End Sub

See also
- BarGraph Object (Page 1914)
- TypeLimitLow5 Property (Page 2455)
- CheckLimitLow5 Property (Page 2175)

LimitMax Property

Description

Defines or returns the high limit value as an absolute value dependent on the data format in the case of the IOField object.

If the value to be displayed exceeds the upper limit value, it is identified by a series of "***", indicating it cannot be displayed.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the high limit for a decimal value will be set to "100":

```vba
Sub IOFieldConfiguration()
  'VBA605
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .DataFormat = 1
    .LimitMax = 100
  End With
End Sub
```

See also

DataFormat Property (Page 2210)
IOField Object (Page 1992)

LimitMin Property

Description

Defines or returns the low limit value as an absolute value dependent on the data format in the case of the IOField object.

If the value to be displayed exceeds the upper limit value, it is identified by a series of *** , indicating it cannot be displayed.

Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example the high limit for a decimal value will be set to "0":

```vba
Sub IOFieldConfiguration()
  'VBA606
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .DataFormat = 1
    .LimitMin = 0
  End With
End Sub
```
LineJoinStyle property

Description

Defines the way that corners are displayed in a tube polygon.

- **Angle** The tubes are joined at corner points without rounding.
- **Round** The tubes are rounded at the outside corner points.

Example

```
Sub TextListConfiguration()
'VBA607
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
  .ListType = 0
End With
End Sub
```

ListType Property

Description

Defines or returns the list type in the case of the TextList object. Value range from 0 to 2.

<table>
<thead>
<tr>
<th>List type</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>0</td>
</tr>
<tr>
<td>Binary</td>
<td>1</td>
</tr>
<tr>
<td>Bit</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the list type will be set to "Decimal":

```
Sub TextListConfiguration()
'VBA607
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
  .ListType = 0
End With
End Sub
```
See also

TextList Object (Page 2076)

LockBackColor Property

Description

Defines or returns the background color of the button for a locked measuring point in the case of the GroupDisplay object. LONG write-read access.

The LockStatus property must be set to TRUE for the background color to be displayed.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color for a locked measuring point will be set to "Red":

```vba
Sub GroupDisplayConfiguration()
'VBA608
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
.LockStatus = True
.LockBackColor = RGB(255, 0, 0)
End With
End Sub
```

See also

LockStatus Property (Page 2315)

GroupDisplay Object (Page 1980)
LockedByCreatorID Property

Description

TRUE, if a picture was created and/or referenced by SIMATIC Manager. BOOLEAN read access.

If a picture was created in SIMATIC Manager, you may process and subsequently save it in WinCC. You may, however, not delete this picture in WinCC. SIMATIC Manager administers a code for each picture, the so-called CreatorID, which cannot be changed in WinCC.

You may process the picture in WinCC, however, overwriting the picture with a WinCC picture (LockedByCreatorID = FALSE) will be prevented. This may be checked by examining the LockedByCreatorID property of an existing file prior to writing during the SaveAs method. If such a picture is saved into a new (not yet existing) or an existing WinCC picture using the SaveAs method, the CreatorID will not be passed on.

Example 1

In the following example, a picture created with SIMATIC Manager (LockedByCreatorID = TRUE) is opened, processed, and saved. The value of the LockedByCreatorID property is not changed.

Sub SaveDocAs_1()
'VBA810
'open an existing file, change it and save it
Dim docOld As Document
Const strFile As String = "Simatic_001.Pdl"
',
Set docOld = Application.Documents.Open(Application.ApplicationDataPath & strFile,
hmiOpenDocumentTypeInvisible)
docOld.Width = docOld.Width + 1
docOld.Save
',
MsgBox "LockedByCreatorID = " & docOld.LockedByCreatorID, vbOKOnly, "Result"
',
docOld.Close
Set docOld = Nothing
',
End Sub
Example 2

In this example, a new picture is saved as a new file using the SaveAs method. To check if the picture is permitted to be saved, the LockedByCreatorID property is checked. In the new file the LockedByCreator property is reset.

```vba
Sub SaveDocAs_2()
'VBA811
'create a new file and overwrite it to an existing file,
'if it is not 'locked by CreatorID'
Dim docNew As Document
Dim docOld As Document
Const strFile As String = "Simatic_001.Pdl"

Set docNew = Application.Documents.Add(hmiOpenDocumentTypeInvisible)

If docOld.LockedByCreatorID = False Then
    docOld.Close
    docNew.SaveAs(Application.ApplicationDataPath & strFile)
Else
    MsgBox "File cannot be stored (LockedByCreatorID). ", vbOKOnly, "Result"
End If

docOld.Close
docNew.Close
Set docOld = Nothing
Set docNew = Nothing
End Sub
```

See also

- SaveAs Method (Page 1880)
- Document Object (Page 1949)

LockStatus Property

Description

TRUE if a locked measuring point is to be displayed with the Object GroupDisplay. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color for a locked measuring point will be set to "Red":

```vba
Sub GroupDisplayConfiguration()
'VBA609
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
.LockStatus = True
.LockBackColor = RGB(255, 0, 0)
End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

LockText Property

Description

Defines the button labels for a locked measuring point in the case of the GroupDisplay object. The LockStatus property must be set to TRUE for the label to be displayed.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the label for a locked measuring point will be set to "Locked":

```vba
Sub GroupDisplayConfiguration()
'VBA610
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
.LockStatus = True
.LockText = "gesperrt"
End With
End Sub
```
See also

LockStatus Property (Page 2315)
GroupDisplay Object (Page 1980)

LockTextColor Property

Description

Defines or returns the color of the button label for a locked measuring point in the case of the GroupDisplay object. LONG write-read access.

The LockStatus property must be set to TRUE for the background color to be displayed.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the button label for a locked measuring point will be set to "Yellow":

Sub GroupDisplayConfiguration()
  'VBA611
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .LockStatus = True
    .LockTextColor = RGB(0, 255, 255)
  End With
End Sub

See also

LockStatus Property (Page 2315)
GroupDisplay Object (Page 1980)
LongStrokesBold Property

Description
TRUE if the long strokes on the scale of the BarGraph object are to be displayed in bold. BOOLEAN write-read access.

Example:
The "BarGraphConfiguration()" procedure configures In this example the long strokes will not be displayed in bold:

```vba
Sub BarGraphConfiguration()
  'VBA612
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    LongStrokesBold = False
  End With
End Sub
```

See also
BarGraph Object (Page 1914)

LongStrokesOnly Property

Description
TRUE if just the long strokes on the scale of the BarGraph object are to be displayed. BOOLEAN write-read access.

Example:
The "BarGraphConfiguration()" procedure configures In this example, only the long strokes will be displayed:

```vba
Sub BarGraphConfiguration()
  'VBA613
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    LongStrokesOnly = True
  End With
End Sub
```
See also

BarGraph Object (Page 1914)

LongStrokesSize Property

**Description**

The "BarGraphConfiguration()" procedure configures

**Example:**

In this example the length of the axis section strokes will be set to "10".

```vba
Sub BarGraphConfiguration()
    'VBA614
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .LongStrokesSize = 10
    End With
End Sub
```

See also

AxisSection Property (Page 2130)
BarGraph Object (Page 1914)

LongStrokesTextEach Property

**Description**

Defines or returns which strokes will be labeled when displaying the scale on the BarGraph object (1 = every stroke, 2 = every second stroke, etc.).
Example:

The "BarGraphConfiguration()" procedure configures In this example every third stroke will be labeled:

```vba
Sub BarGraphConfiguration()
'VBA615
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
LongStrokesTextEach = 3
End With
End Sub
```

See also

BarGraph Object (Page 1914)

M

Macro Property

Description

For a user-defined menu entry or icon, defines the VBA macro that will be executed upon selection.

Example:

In the following example, a user-defined menu with two menu entries is created, which retrieve two different VBA macros:

```vba
Sub CreateDocumentMenusUsingMacroProperty()
'VBA616
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "My first menuitem")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second menuitem")
' 'To assign a macro to every menuitem:
With ActiveDocument.CustomMenus("DocMenu1")
 .MenuItems("dmItem1_1").Macro = "TestMacro1"
 .MenuItems("dmItem1_2").Macro = "TestMacro2"
End With
End Sub
```
You can call the following two procedures via the menu items in the user-defined menu "DocMenu1":

Sub TestMacro1()
    MsgBox "TestMacro1 is executed"
End Sub

Sub TestMacro2()
    MsgBox "TestMacro2 is executed"
End Sub

See also
 ToolbarItem Object (Page 2083)
 MenuItem Object (Page 2013)
 How to assign VBA macros to menus and toolbars (Page 1642)

Marker Property

Description
 TRUE if the limit values are to be displayed as a scale value in the case of the BarGraph object.
 BOOLEAN write-read access.

Example:

The "BarGraphConfiguration()" procedure configures In this example, the limit values will be displayed as scale values:

Sub BarGraphConfiguration()
    'VBA617
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .Marker = True
    End With
End Sub

See also
 BarGraph Object (Page 1914)
Max Property

Description

Defines or returns the absolute value in the case of a full value display.
This value is displayed if the scale display is active.

Example:

The "BarGraphConfiguration()" procedure configures In this example the absolute value will be set to "10".

Sub BarGraphConfiguration()
'VBA618
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjec.ts.AddHMIObjec.t("Bar1", "HMIBarGraph")
With objBarGraph
  .Max = 10
End With
End Sub

See also

Slider object (Page 2062)
BarGraph Object (Page 1914)
3DBarGraph Object (Page 1894)

MaxIndex property

Description

Shows the highest index of all configurable alarm and status combinations at the "HMIAdvancedStateDisplay" object.

MaximizeButton Property

Description

TRUE if the ApplicationWindow object can be maximized in Runtime. BOOLEAN write-read access.
Example:

The "ApplicationWindowConfig" procedure accesses the properties of the application window. In this example the application window will receive a Maximize button in Runtime:

```vba
Sub ApplicationWindowConfig()
    'VBA619
    Dim objAppWindow As HMIApplicationWindow
    Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow1", "HMIApplicationWindow")
    With objAppWindow
        .MaximizeButton = True
    End With
End Sub
```

See also

ApplicationWindow Object (Page 1911)

MaxZoom Property

Description

Defines or returns the maximum zoom level for the layer.

Example:

The "LayerInfo()" procedure outputs the name and zoom configuration for each layer of the current picture:

```vba
Sub LayerInfo()
    'VBA620
    Dim colLayers As HMILayers
    Dim objSingleLayer As HMILayer
    Dim iAnswer As Integer
    Set colLayers = ActiveDocument.Layers
    For Each objSingleLayer In colLayers
        With objSingleLayer
        End With
        If vbCancel = iAnswer Then Exit For
        Next objSingleLayer
        End Sub
```
MCGUBackColorOff-Eigenschaft

Description
In the case of the GroupDisplay object, defines or returns the background color for the "Went Out Unacknowledged" status when the flash status is "Off". LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "Off" will be set to "Red":

```vba
Sub GroupDisplayConfiguration()
'VBA621
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
 .MCGUBackColorOff = RGB(255, 0, 0)
End With
End Sub
```

See also
GroupDisplay Object (Page 1980)

MCGUBackColorOn Property

Description
In the case of the GroupDisplay object, defines or returns the background color for the "Went Out Unacknowledged" status when the flash status is "On". LONG write-read access.
Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "On" will be set to "White":

```vba
Sub GroupDisplayConfiguration()
  'VBA622
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .MCGUBackColorOn = RGB(255, 255, 255)
  End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCGUBackFlash Property

Description

TRUE if the background to the GroupDisplay object is to flash when a message goes out unacknowledged. BOOLEAN write-read access.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color is to flash when a message goes out unacknowledged:

```vba
Sub GroupDisplayConfiguration()
  'VBA623
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .MCGUBackFlash = True
  End With
End Sub
```
MCGUTextColorOff Property

Description

In the case of the GroupDisplay object, defines or returns the text color for the "Went Out Unacknowledged" status when the flash status is "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "Off" will be set to "Blue":

```vba
Sub GroupDisplayConfiguration()
'VBA624
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .MCGUTextColorOff = RGB(0, 0, 255)
End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCGUTextColorOn Property

Description

In the case of the GroupDisplay object, defines or returns the background color to the text for the "Went Out Unacknowledged" status when the flash status is "On". LONG write-read access.
Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "On" will be set to "Black":

```vba
Sub GroupDisplayConfiguration()
  'VBA625
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", 
                  "HMIGroupDisplay")
  With objGroupDisplay
      .MCGUTextColorOn = RGB(0, 0, 0)
  End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCGUTextFlash Property

Description

TRUE if the font for the GroupDisplay object is to flash when a message goes out unacknowledged. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the font is to flash when a message goes out unacknowledged:

```vba
Sub GroupDisplayConfiguration()
    'VBA626
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCGUTextFlash = True
    End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKOBackColorOff Property

Description

In the case of the GroupDisplay object, defines or returns the background color for the "Came In" status when the flash status is "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "Off" will be set to "Red":

```vba
Sub GroupDisplayConfiguration()
    'VBA627
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKOBackColorOff = RGB(255, 0, 0)
    End With
End Sub
```
See also

GroupDisplay Object (Page 1980)

MCKOBBackColorOn Property

Description

In the case of the GroupDisplay object, defines or returns the background color for the "Came In" status when the flash status is "On". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "On" will be set to "White":

```vba
Sub GroupDisplayConfiguration()
 'VBA628
 Dim objGroupDisplay As HMIGroupDisplay
 Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIOBJECT("GroupDisplay1", "HMIGroupDisplay")
 With objGroupDisplay
 .MCKOBBackColorOn = RGB(255, 255, 255)
 End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKOBBackFlash Property

Description

TRUE if the background to the GroupDisplay object is to flash when a message goes out unacknowledged. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "Off" will be set to "Blue":

```vba
Sub GroupDisplayConfiguration()
    'VBA630
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKOTextColorOff = RGB(0, 0, 255)
    End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKOTextColorOff Property

Description

In the case of the GroupDisplay object, defines or returns the text color for the "Came In" status when the flash status is "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "Off" will be set to "Blue":

```vba
Sub GroupDisplayConfiguration()
    'VBA629
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKOBackFlash = True
    End With
End Sub
```
See also
GroupDisplay Object (Page 1980)

MCKOTextColorOn Property

Description
In the case of the GroupDisplay object, defines or returns the background color to the text for the "Came In" status when the flash status is "On". LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "On" will be set to "Black":

```
Sub GroupDisplayConfiguration()
  'VBA631
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .MCKOTextColorOn = RGB(0, 0, 0)
  End With
End Sub
```

See also
GroupDisplay Object (Page 1980)

MCKOTextFlash Property

Description
TRUE if the font for the GroupDisplay object is to flash when a message goes out unacknowledged. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the font is to flash when a message goes out unacknowledged:

Sub GroupDisplayConfiguration()
    'VBA632
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKOTextFlash = True
    End With
End Sub

See also

GroupDisplay Object (Page 1980)

MCKQBackColorOff Property

Description

In the case of the GroupDisplay object, defines or returns the background color for the "Went Out Acknowledged" status when the flash status is "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "Off" will be set to "Red":

Sub GroupDisplayConfiguration()
    'VBA633
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKQBackColorOff = RGB(255, 0, 0)
    End With
End Sub
See also

GroupDisplay Object (Page 1980)

MCKQBackColorOn Property

Description
In the case of the GroupDisplay object, defines or returns the background color for the "Went Out Acknowledged" status when the flash status is "On". LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color when the flash status is "On" will be set to "White":

```vba
Sub GroupDisplayConfiguration()
'VBA634
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjets.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .MCKQBackColorOn = RGB(255, 255, 255)
End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKQBackColorFlash Property

Description
TRUE if the background to the GroupDisplay object is to flash when a message goes out acknowledged. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color is to flash when a message goes out unacknowledged:

```vba
Sub GroupDisplayConfiguration()
    'VBA635
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKQBackFlash = True
    End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKQTextColorOff Property

Description

In the case of the GroupDisplay object, defines or returns the text color for the "Went Out Acknowledged" status when the flash status is "Off". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "Off" will be set to "Blue":

```vba
Sub GroupDisplayConfiguration()
    'VBA636
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .MCKQTextColorOff = RGB(0, 0, 255)
    End With
End Sub
```
See also

GroupDisplay Object (Page 1980)

MCKQTextColorOn Property

Description

In the case of the GroupDisplay object, defines or returns the background color to the text for the "Went Out Acknowledged" status when the flash status is "On". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color to the text when the flash status is "On" will be set to "Black":

```vba
Sub GroupDisplayConfiguration()
'VBA637
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
 .MCKQTextColorOn = RGB(0, 0, 0)
End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCKQTextFlash Property

Description

TRUE if the font for the GroupDisplay object is to flash when a message goes out acknowledged. BOOLEAN write-read access.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. 
In this example the font is to flash when a message goes out unacknowledged:

```
Sub GroupDisplayConfiguration()
'VBA638
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .MCKQTextFlash = True
End With
End Sub
```

See also

GroupDisplay Object (Page 1980)

MCText Property

Description

Defines or returns the label for the appropriate message class in the case of the GroupDisplay object.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. 
In this example the label for the "Alarm High" message class will be set to "Alarm High":

```
Sub GroupDisplayConfiguration()
'VBA639
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .MessageClass = 0
  .MCText = "Alarm High"
End With
End Sub
```
See also

MessageClass Property (Page 2339)
GroupDisplay Object (Page 1980)

MenuItems Property

Description

Returns a listing containing all the menu entries in the user-defined menu.

Example:

The "CreateMenuItem()" procedure creates the "Delete Objects" menu and adds two menu entries ("Delete Rectangles" and "Delete Circles"). In this example the labels will then be output:

```vba
Sub CreateMenuItem()
    'VBA640
    Dim objMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem
    Dim iIndex As Integer
    iIndex = 1
    'Add new menu "Delete objects" to the menubar:
    Set objMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DeleteObjects", "Delete objects")
    'Add two menuitems to menu "Delete objects"
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(1, "DeleteAllRectangles", "Delete rectangles")
    Set objMenuItem = objMenu.MenuItems.InsertMenuItem(2, "DeleteAllCircles", "Delete circles")
    'Output label of menu:
    MsgBox ActiveDocument.CustomMenus(1).Label
    'Output labels of all menuitems:
    For iIndex = 1 To objMenu.MenuItems.Count
        MsgBox objMenu.MenuItems(iIndex).Label
    Next iIndex
End Sub
```

See also

Menu Object (Page 2009)
MenuItem Object (Page 2013)
MenuItemType Property

Description

Returns the type for a user-defined menu entry. Read only access.

<table>
<thead>
<tr>
<th>Returned Value</th>
<th>Type of Menu Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Separator (Separator)</td>
</tr>
<tr>
<td>1</td>
<td>Submenu (SubMenu)</td>
</tr>
<tr>
<td>2</td>
<td>Menu Entry (MenuItem)</td>
</tr>
</tbody>
</table>

Example:

The "ShowMenuTypes()" procedure outputs the types for the menu entries in the first user-defined menu:

```vba
Sub ShowMenuTypes()
  'VBA641
  Dim iMaxMenuItems As Integer
  Dim iMenuItemType As Integer
  Dim strMenuItemType As String
  Dim iIndex As Integer
  iMaxMenuItems = ActiveDocument.CustomMenus(1).MenuItems.Count
  For iIndex = 1 To iMaxMenuItems
    iMenuItemType = ActiveDocument.CustomMenus(1).MenuItems(iIndex).MenuItemType
    Select Case iMenuItemType
      Case 0
        strMenuItemType = "Trennstrich (Separator)"
      Case 1
        strMenuItemType = "Untermenü (SubMenu)"
      Case 2
        strMenuItemType = "Menüeintrag (MenuItem)"
    End Select
    MsgBox iIndex & ". Menuitemtype: " & strMenuItemType
  Next iIndex
End Sub
```

See also

MenuItem Object (Page 2013)
Menu Object (Page 2009)
MenuToolBarConfig Property

Description
Specifies the configuration file with the user-defined menu and toolbars for the "HMIPictureWindow" object or returns the name of the configuration file. STRING write-read access.

MessageClass Property

Description
Specifies the respective message type (Alarm High, Alarm Low, Warning High, Warning Low, etc.) for which the attribute settings "Display Text", "Came In", "Came In Acknowledged" and "Went Out Unacknowledged" are configured for the "GroupDisplay" and "AdvancedAnalogDisplay" objects.

<table>
<thead>
<tr>
<th>MessageClass</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmHigh</td>
<td>0</td>
</tr>
<tr>
<td>AlarmLow</td>
<td>1</td>
</tr>
<tr>
<td>WarningHigh</td>
<td>2</td>
</tr>
<tr>
<td>WarningLow</td>
<td>3</td>
</tr>
<tr>
<td>Tolerance High</td>
<td>4</td>
</tr>
<tr>
<td>Tolerance Low</td>
<td>5</td>
</tr>
<tr>
<td>AS Control System Fault</td>
<td>6</td>
</tr>
<tr>
<td>AS Process Control Error</td>
<td>7</td>
</tr>
<tr>
<td>OS Process Control Error</td>
<td>8</td>
</tr>
</tbody>
</table>

Example
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the background color for the "AlarmHigh" message type when the flash status is "Off" will be set to "Red":

```vba
Sub GroupDisplayConfiguration()
  'VBA642
  Dim objGroupDisplay As HMIGroupDisplay
  Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
  With objGroupDisplay
    .MessageClass = 0
    .MCGUBackColorOff = RGB(255, 0, 0)
  End With
End Sub
```
See also

GroupDisplay Object (Page 1980)

Min Property

Description

Defines or returns the absolute value in the case of the smallest value display. This value is displayed if the scale display is active.

Example:

The "BarGraphConfiguration()" procedure configures In this example the absolute value will be set to "1".

Sub BarGraphConfiguration()
  'VBA643
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjetcts.AddHMIObjetct("Bar1", "HMIBarGraph")
  With objBarGraph
    .Min = 1
  End With
End Sub

See also

Slider object (Page 2062)
BarGraph Object (Page 1914)
3DBarGraph Object (Page 1894)

MinZoom Property

Description

Defines or returns the minimum zoom level for the layer.
Example:

The "LayerInfo()" procedure outputs the name and zoom configuration for each layer of the current picture:

Sub LayerInfo()
    'VBA644
    Dim colLayers As HMILayers
    Dim objLayer As HMIayer
    Dim strMaxZoom As String
    Dim strMinZoom As String
    Dim strLayerName As String
    Dim iAnswer As Integer
    Set colLayers = ActiveDocument.Layers
    For Each objLayer In colLayers
        With objLayer
            strMinZoom = .MinZoom
            strMaxZoom = .MaxZoom
            strLayerName = .Name
            iAnswer = MsgBox("Layername: " & strLayerName & vbCrLf & "Min. zoom: " & strMinZoom & vbCrLf & "Max. zoom: " & strMaxZoom, vbOKCancel)
        End With
        If vbCancel = iAnswer Then Exit For
    Next objLayer
End Sub

See also

Layer Object (Page 2001)
Editing Layers with VBA (Page 1657)

Modified Property

Description
TRUE if the source code for a script or picture has been changed. BOOLEAN read access.
Example:
In the following example a check will be made on whether the active picture has been changed:

Sub CheckModificationOfActiveDocument()
'VBA645
Dim strCheck As String
Dim bModified As Boolean
bModified = ActiveDocument.Modified
Select Case bModified
Case True
strCheck = "Active document is modified"
Case False
strCheck = "Active document is not modified"
End Select
MsgBox strCheck
End Sub

See also
ScriptInfo Object (Page 2058)
Document Object (Page 1949)

Moveable Property

Description
TRUE if the ApplicationWindow and PictureWindow objects can be moved in Runtime.
BOOLEAN write-read access.

Example:
The "ApplicationWindowConfig" procedure accesses the properties of the application window.
In this example it shall be possible to move the application window in Runtime:

Sub ApplicationWindowConfig()
'VBA646
Dim objAppWindow As HMIApplicationWindow
Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow1",
"HMIApplicationWindow")
With objAppWindow
.Moveable = True
End With
End Sub
See also

PictureWindow Object (Page 2029)
ApplicationWindow Object (Page 1911)

N-O

Name Property

Description

Returns the name of the object. STRING read access.

Example:

The "LayerInfo()" procedure outputs the name and zoom configuration for each layer of the current picture:

Sub LayerInfo()
'VBA647
Dim colLayers As HMILayers
Dim objLayer As HMILayer
Dim strMaxZoom As String
Dim strMinZoom As String
Dim strLayerName As String
Dim iAnswer As Integer
Set colLayers = ActiveDocument.Layers
For Each objLayer In colLayers
With objLayer
strMinZoom = .MinZoom
strMaxZoom = .MaxZoom
strLayerName = .Name
iAnswer = MsgBox("Layername: " & strLayerName & vbCrLf & "Min. zoom: " & strMinZoom & vbCrLf & "Max. zoom: " & strMaxZoom, vbOKCancel)
End With
If vbCancel = iAnswer Then Exit For
Next objLayer
End Sub

See also

Trigger Object (Page 2087)
SymbolLibrary Object (Page 2074)
Property Object (Page 2042)
HMIObject Object (Page 1988)
Layer Object (Page 2001)
FolderItem Object (Page 1972)
Document Object (Page 1949)
Application Object (Page 1909)

Name Property (FolderItem)

Description

Returns the internal name of the specified object of the "FolderItem" type. Read only access.

Example:

In this example the internal name is output for the "PC" object contained in the Global Components Library:

Sub ShowInternalNameOfFolderItem()
    'VBA536
    Dim objGlobalLib As HMISymbolLibrary
    Set objGlobalLib = Application.SymbolLibraries(1)
    MsgBox objGlobalLib.FolderItems(2).Folder(2).Folder.Item(1).Name
End Sub

See also

FolderItem Object (Page 1972)
Accessing the component library with VBA (Page 1646)

NegativeValue Property

Description

Use the BinaryResultInfo property to return the BinaryResultInfo object.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and the associated property values will be assigned to both the binary value ranges:

```vba
Sub AddDynamicDialogToCircleRadiusTypeBinary()
    'VBA648
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_C", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeBool
        .BinaryResultInfo.NegativeValue = 20
        .BinaryResultInfo.PositiveValue = 40
    End With
End Sub
```

See also
- VBA Reference (Page 1745)
- PositiveValue Property (Page 2380)
- BinaryResultInfo Object (Page 1918)

**NibbleSelect property**

**Description**

Only used internally.

See also
- AdvancedStateDisplay object (Page 1905)

**Number Property**

**Description**

Returns the layer number of a "Layer" type object. The counting starts with 1. The first layer, "Layer0", returns the value "0". READ access.
Example:

This example outputs the name, number and index of a layer:

```vba
Sub ShowLayerWithNumbers()
'VBA803
Dim colLayers As HMILayers
Dim objLayer As HMILayer
Dim iAnswer As Integer
Dim iIndex As Integer
iIndex = 1
Set colLayers = ActiveDocument.Layers
For Each objLayer In colLayers
    iAnswer = MsgBox("Layername: " & objLayer.Name & vbCrLf & "Layernumber: " & objLayer.Number & vbCrLf & "Layersindex: " & iIndex, vbOKCancel)
    iIndex = iIndex + 1
Next objLayer
End Sub
```

See also

Layer Object (Page 2001)

NumberLines Property

Description

**TextList**

Defines for the "TextList object" how many lines the selection list should contain or returns the value. If the configured lines with their number, font size and font do not fit into the dimensions of the object, a vertical scroll bar is added to the selection list.

**Combo box and list box**

Defines or returns the number of lines of text for the "Combo box" and "List box" objects. You can define a maximum of 32,000 lines.

At the same time, the value of the "Number of rows" attribute specifies the high limit value for the "Index" attribute in the "Font" property group. Changing the value can have the following effects:

- Increasing the number: New lines are added at the bottom. The default labeling of the new file can be changed using the "Text" attribute in the "Font" property group.
- Reducing the number: All lines are removed for which the value of the "Index" attribute is higher than the new number.
Example

The "TextListConfiguration()" procedure accesses the properties of the "TextList" object. In this example a selection list is created and the number of visible lines is set to three:

```vba
Sub TextListConfiguration()
'VBA649
Dim objTextList As HMITextList
' 'Insert new TextList in current picture:
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
  .NumberLines=3
End With
End Sub
```

See also

TextList Object (Page 2076)

ObjectName Property

Description

Depending on the source and destination object types for the direct connection, either defines or returns the name of the constant, object or tag.

The two tables show you when you must use the ObjectName property. A "--" means that the property is assigned an empty string (""") by default when the DirectConnection object is created.

Source object type (SourceLink Property)

<table>
<thead>
<tr>
<th>Type Property</th>
<th>AutomationName Property</th>
<th>ObjectName Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmiSourceTypeConstant</td>
<td>--</td>
<td>Name of the constant (e.g. the picture name)</td>
</tr>
<tr>
<td>hmiSourceTypeProperty</td>
<td>Property of the source object (e.g. &quot;Top&quot;)</td>
<td>Name of the source object (e.g. &quot;Rectangle_A&quot;)</td>
</tr>
<tr>
<td>hmiSourceTypePropertyOfThisObject</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>hmiSourceTypeVariableDirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiSourceTypeVariableIndirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
</tbody>
</table>

Destination object type (DestinationLink Property)

<table>
<thead>
<tr>
<th>Type Property</th>
<th>AutomationName Property</th>
<th>ObjectName Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmiDestTypeProperty</td>
<td>Property of the destination object (e.g. &quot;Left&quot;)</td>
<td>Name of the destination object (e.g. &quot;Rectangle_A&quot;)</td>
</tr>
<tr>
<td>hmiDestTypePropertyOfThisObject</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Type Property</td>
<td>AutomationName Property</td>
<td>ObjectName Property</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>hmiDestTypePropertyOfActualWindow</td>
<td>Property of the destination object (e.g. &quot;Left&quot;)</td>
<td>--</td>
</tr>
<tr>
<td>hmiDestTypeVariableDirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeVariableIndirect</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeDirectMessage</td>
<td>--</td>
<td>Tag name</td>
</tr>
<tr>
<td>hmiDestTypeIndirectMessage</td>
<td>--</td>
<td>Tag name</td>
</tr>
</tbody>
</table>
Example:

In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
'VBA650
Dim objButton As HMIButton
Dim objRectangleA As HMIRectangle
Dim objRectangleB As HMIRectangle
Dim objEvent As HMIEvent
Dim objDirConnection As HMIDirectConnection
Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objRectangleA
  .Top = 100
  .Left = 100
End With
With objRectangleB
  .Top = 250
  .Left = 400
  .BackColor = RGB(255, 0, 0)
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 100
  .Text = "SetPosition"
End With

'Directconnection is initiated by mouseclick:
Set objDirConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
With objDirConnection
  'Sourceobject: Property "Top" of Rectangle_A
  .SourceLink.Type = hmiSourceTypeProperty
  .SourceLink.ObjectName = "Rectangle_A"
  .SourceLink.AutomationName = "Top"
  ,
  'Targetobject: Property "Left" of Rectangle_B
  .DestinationLink.Type = hmiDestTypeProperty
  .DestinationLink.ObjectName = "Rectangle_B"
  .DestinationLink.AutomationName = "Left"
End With
End Sub

See also

Type Property (Page 2450)
SourceLink Property (Page 2422)
DestinationLink Property (Page 2211)
### ObjectSizeDecluttering Property

**Description**

TRUE, if objects of the specified picture outside of two configured sizes are to be faded out. BOOLEAN write-read access.

Define the size range with the aid of the SetDeclutterObjectSize method.

**Example:**

In the following example the settings for the lowest layer are configured in the active picture:

```vba
Sub ConfigureSettingsOfLayer()
    'VBA651
    Dim objLayer As HMILayer
    Set objLayer = ActiveDocument.Layers(1)
    With objLayer
        .MinZoom = 10
        .MaxZoom = 100
        .Name = "Configured with VBA"
    End With
    'Define fade-in and fade-out of objects:
    With ActiveDocument
        .LayerDecluttering = True
        .ObjectSizeDecluttering = True
        .SetDeclutterObjectSize 50, 100
    End With
End Sub
```

**See also**

- Document Object (Page 1949)
- Editing Layers with VBA (Page 1657)

### OffsetLeft Property

**Description**

Defines or returns the distance of the picture from the left edge of the picture window.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured

Sub PictureWindowConfig()
'VBA652
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10
  'Replace the picturename "Test.PDL" with the name of
  'an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ServerPrefix = ""
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub

See also

PictureWindow Object (Page 2029)

OffsetTop Property

Description

Defines or returns the distance of the picture from the top edge of the picture window.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured.

```vba
Sub PictureWindowConfig()
  Dim objPicWindow As HMIPictureWindow
  Set objPicWindow = ActiveDocument.HMIOObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
  With objPicWindow
    .AdaptPicture = False
    .AdaptSize = False
    .Caption = True
    .CaptionText = "Picturewindow in runtime"
    .OffsetLeft = 5
    .OffsetTop = 10
    .PictureName = "Test.PDL"
    .ScrollBars = True
    .ServerPrefix = ""
    .TagPrefix = "Struct."
    .UpdateCycle = 5
    .Zoom = 100
  End With
End Sub
```

See also

PictureWindow Object (Page 2029)

OnTop Property

Description

TRUE if the ApplicationWindow object is always in the foreground in Runtime. BOOLEAN write-read access.
Example:

The "ApplicationWindowConfig" procedure accesses the properties of the application window. In this example the application window will always be in the foreground in Runtime:

```vba
Sub ApplicationWindowConfig()
'VBA654
Dim objAppWindow As HMIApplicationWindow
Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow1", "HMIApplicationWindow")
With objAppWindow
  .OnTop = True
End With
End Sub
```

See also

ApplicationWindow Object (Page 1911)

Operation Property

Description

TRUE if the object can be used or operated in Runtime. BOOLEAN write-read access.
Example:

In this example the status of the operator-control enables will be output for all objects in the active picture:

Sub ShowOperationStatusOfAllObjects()
  'VBA655
  Dim objObject As HMIObject
  Dim bStatus As Boolean
  Dim strStatus As String
  Dim strName As String
  Dim iMax As Integer
  Dim iIndex As Integer
  Dim iAnswer As Integer
  iMax = ActiveDocument.HMIObjects.Count
  iIndex = 1
  For iIndex = 1 To iMax
      strName = ActiveDocument.HMIObjects(iIndex).ObjectName
      bStatus = ActiveDocument.HMIObjects(iIndex).Operation
      Select Case bStatus
            Case True
                strStatus = "yes"
            Case False
                strStatus = "no"
      End Select
      iAnswer = MsgBox("Object: " & strName & vbCrLf & "Operator-Control enable: " & strStatus, vbOKCancel)
      If vbCancel = iAnswer Then Exit For
  Next iIndex
  If 0 = iMax Then MsgBox "No objects in the active document."
End Sub

See also

HMIObject Object (Page 1988)
Document Object (Page 1949)

OperationMessage Property

Description

TRUE, if a message should be output upon successful operation. The reason for the operation can only be input if the "OperationReport" property is set to "True". BOOLEAN write-read access.

The operation is sent to the message system, and is archived. Using the message system, a message may be output in a message line, for example.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example, an operation is supposed to be sent to the message system:

```
Sub IOFieldConfiguration()
  Dim objIOField As HMIIOField
  Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  With objIOField
    .OperationReport = True
    .OperationMessage = True
  End With
End Sub
```

See also

- OperationReport Property (Page 2355)
- TextList Object (Page 2076)
- Slider object (Page 2062)
- OptionGroup Object (Page 2025)
- IOField Object (Page 1992)
- CheckBox Object (Page 1924)

OperationReport Property

Description

TRUE, if the reason for an operation should be recorded. BOOLEAN write-read access.

When the object is used or operated in Runtime, a dialog opens in which the operator can input the reason for the operation in the form of text. The operation is sent to the message system, and is archived.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example, an operation is supposed to be sent to the message system:

```vba
Sub IOFieldConfiguration()
    'VBA657
    Dim objIOField As HMIIOField
    Set objIOField = ActiveDocument.HMIOObjects.AddHMIOObject("IOField1", "HMIIOField")
    With objIOField
        .OperationReport = True
        .OperationMessage = True
    End With
End Sub
```

See also

- OperationMessage Property (Page 2354)
- TextList Object (Page 2076)
- Slider object (Page 2062)
- OptionGroup Object (Page 2025)
- IOField Object (Page 1992)
- CheckBox Object (Page 1924)

Orientation Property

Description

TRUE, when the text in the object should be displayed horizontally. BOOLEAN write-read access.

Note

It is only the text that is displayed either horizontally or vertically. The position of the object remains unchanged.
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the text will be displayed vertically:

```vba
Sub ButtonConfiguration()
    'VBA658
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
    With objButton
        .Width = 150
        .Height = 150
        .Text = "Text is displayed vertical"
        .Orientation = False
    End With
End Sub
```

See also

TextList Object (Page 2076)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
IOField Object (Page 1992)
CheckBox Object (Page 1924)
Button Object (Page 1921)

OriginalPropertyName property

Description

Only used internally.

See also

FaceplateObjects object (Page 1971)

OutputFormat Property

Description

Defines how the output value shall be displayed, or returns the set value. The representation is dependent on the data format.
Example:

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example, data type "Decimal" will be set for the I/O field: The output value will be displayed with two decimals and three digits to the right of the decimal point:

Sub IOFieldConfiguration()
'VBA659
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.DataFormat = 1
.OutputFormat = "99,999"
End With
End Sub

See also

DataFormat Property (Page 2210)
IOField Object (Page 1992)

OutputValue Property

Description

Defines or returns presetting for the value to be displayed.

This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.

Example

The "IOFieldConfiguration()" procedure accesses the properties of the I/O field. In this example, the output value is set to "0.000":

Sub IOFieldConfiguration()
'VBA660
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.OutputValue = "0"
End With
End Sub
See also

TextList Object (Page 2076)
IOField Object (Page 1992)

OutputValue property

Description
Specifies the interconnection with any analog / text tag. The analog display represents the value of this tag in the configured colors depending on the alarm state.

P-Q

PaintColor_QualityCodeBad property

Description
Defines the color in which the grid is shown when a poor status exists, for example, if the connection to the server is interrupted.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

PaintColor_QualityCodeUnCertain property

Description
Defines the color with which the grid is shown in an uncertain status.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Parent Property

Description
Returns the higher-ranking object in the specified object. Read only access.
Example:

In the following example a copy of the active picture is created and the name of the picture is then output with the aid of the Parent property:

Sub ExampleForParent()
    'VBA661
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    MsgBox objView.Parent.Name
End Sub

See also

Toolbars Object (Listing) (Page 2081)
Menu Object (Page 2009)
Document Object (Page 1949)
Views Object (Listing) (Page 2103)
View Object (Page 2102)
VariableTriggers Object (Listing) (Page 2101)
VariableTrigger Object (Page 2099)
VariableStateValues Object (Listing) (Page 2097)
VariableStateValue Object (Page 2096)
Trigger Object (Page 2087)
ToolBarItems Object (Listing) (Page 2085)
ToolBarItem Object (Page 2083)
Toolbar Object (Page 2080)
TextList Object (Page 2076)
SymbolLibraries Object (Listing) (Page 2073)
SymbolLibrary Object (Page 2074)
StatusDisplay Object (Page 2070)
StaticText Object (Page 2067)
SourceLink Object (Page 2065)
Slider object (Page 2062)
SelectedObjects object (Listing) (Page 2060)
ScriptInfo Object (Page 2058)
RoundRectangle Object (Page 2055)
RoundButton Object (Page 2052)
Rectangle Object (Page 2049)
Properties Object (Listing) (Page 2041)
Property Object (Page 2042)
PolyLine Object (Page 2038)
Polygon Object (Page 2035)
PictureWindow Object (Page 2029)
PieSegment Object (Page 2032)
OptionGroup Object (Page 2025)
OLEObject Object (Page 2023)
MenuItems Object (Listing) (Page 2015)
Menuitem Object (Page 2013)
Menus Object (Listing) (Page 2011)
Line Object (Page 2004)
Layers Object (Listing) (Page 2003)
Layer Object (Page 2001)
LanguageTexts Object (Listing) (Page 2000)
LanguageText Object (Page 1999)
LanguageFonts Object (Listing) (Page 1997)
LanguageFont Object (Page 1996)
IOField Object (Page 1992)
HMIObjects Object (Listing) (Page 1990)
HMIObject Object (Page 1988)
HMIDefaultObjects Object (Listing) (Page 1985)
GroupedObjects Object (Listing) (Page 1983)
GroupDisplay Object (Page 1980)
Group Object (Page 1978)
GraphicObject Object (Page 1975)
FolderItems Object (Listing) (Page 1973)
FolderItem Object (Page 1972)
Events Object (Listing) (Page 1967)
Event Object (Page 1966)
EllipseSegment Object (Page 1963)
EllipseArc Object (Page 1960)
Ellipse Object (Page 1957)
DynamicDialog Object (Page 1955)
ParentCookie property

Description

Only used internally.

See also

FaceplateProperty object (Page 1971)

PasswordLevel Property

Description

Defines the authorization for operation (e.g. no input or no triggering actions) of the object.

<table>
<thead>
<tr>
<th>PasswordLevel</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;No Access Security&gt;</td>
<td>0</td>
</tr>
<tr>
<td>User Administration</td>
<td>1</td>
</tr>
<tr>
<td>Value input</td>
<td>2</td>
</tr>
<tr>
<td>Process controlling</td>
<td>3</td>
</tr>
<tr>
<td>Picture editing</td>
<td>4</td>
</tr>
</tbody>
</table>
### PasswordLevel

<table>
<thead>
<tr>
<th>PasswordLevel</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen change</td>
<td>5</td>
</tr>
<tr>
<td>Window selection</td>
<td>6</td>
</tr>
<tr>
<td>Hard copy</td>
<td>7</td>
</tr>
<tr>
<td>Confirm alarms</td>
<td>8</td>
</tr>
<tr>
<td>Lock alarms</td>
<td>9</td>
</tr>
<tr>
<td>Free alarms</td>
<td>10</td>
</tr>
<tr>
<td>Message editing</td>
<td>11</td>
</tr>
<tr>
<td>Start archive</td>
<td>12</td>
</tr>
<tr>
<td>Stop archive</td>
<td>13</td>
</tr>
<tr>
<td>Edit archive values</td>
<td>14</td>
</tr>
<tr>
<td>Archive editing</td>
<td>15</td>
</tr>
<tr>
<td>Action editing</td>
<td>16</td>
</tr>
<tr>
<td>Project Manager</td>
<td>17</td>
</tr>
<tr>
<td>Activate remote</td>
<td>1000</td>
</tr>
<tr>
<td>Configure remote</td>
<td>1001</td>
</tr>
<tr>
<td>Just monitor</td>
<td>1002</td>
</tr>
</tbody>
</table>

You must first define the operator authorizations in the User Administrator.

**Example:**

```vba
Sub ShowDocumentPath()
    MsgBox ActiveDocument.Path
End Sub
```

**See also**

HMIObject Object (Page 1988)

**Path Property**

**Description**

Returns the full path of the folder in which the specified picture is stored. Read only access.

**Example:**

In this example the path to the folder of the active picture will be output:
Pathname Property

Description

Returns the internal access path to the Components Library for the specified object of the "FolderItem" type. Read only access.

Example:

In this example the internal access path is output for the "PC" object contained in the Global Components Library:

Sub ShowInternalNameOfFolderItem()
    'VBA664
    Dim objGlobalLib As HMISymbolLibrary
    Set objGlobalLib = Application.SymbolLibraries(1)
    MsgBox objGlobalLib.FolderItems(2).Folder(2).Folder.Item(1).PathName
End Sub

See also

FolderItem Object (Page 1972)
Accessing the component library with VBA (Page 1646)

PdlProtection property

Description

Sets a password for a process picture or faceplate type or deletes the password. Write access.

Note

Significance of the password protection

With the PdlProtection property, you can only assign a password to process pictures or faceplate types to, for example, protect the VBA scripts contained in the pictures against unauthorized access.
Examples

In this example, a password is set for the active picture:

Sub ProtectPicture()
  'VBA854
  ActiveDocument.PdlProtection = "Test123"
End Sub

Password protection for the active picture is removed in this example:

Sub UnprotectPicture()
  'VBA855
  ActiveDocument.PdlProtection = ""
End Sub

Note
Write access only
Read access to the password is prevented due to security reasons.

PicDeactReferenced-Eigenschaft

Description

TRUE if the picture assigned to the "Deactivated" status is to be saved in the RoundButton object. Otherwise, only the associated object reference is saved. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the picture assigned to the "Deactivated" status will be referenced:

Sub RoundButtonConfiguration()
  'VBA665
  Dim objRoundButton As HMIRoundButton
  Set objRoundButton = ActiveDocument.HMIOBJECTS.AddHMIObject("RButton1", "HMIRoundButton")
  With objRoundButton
    .PicDeactReferenced = False
  End With
End Sub
See also

RoundButton Object (Page 2052)

PicDeactTransparent Property

Description

Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Disabled" status should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "PicDeactUseTransColor" property is "True".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Red" assigned in the Bitmap object is to be displayed transparent when in the "Deactivated" status.

Sub RoundButtonConfiguration()
'VBA666
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIOObjects.AddHMIObj("RButton1", "HMIRoundButton")
With objRoundButton
.PicDeactTransparent = RGB(255, 0, 0)
.PicDeactUseTransColor = True
End With
End Sub

See also

PicDeactUseTransColor Property (Page 2367)

RoundButton Object (Page 2052)
PicDeactUseTransColor Property

Description

TRUE, when the transparent color defined by the "PicDeactTransparent" property for the "Disable" status should be used. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Red" assigned in the Bitmap object is to be displayed transparent when in the "Deactivated" status:

Sub RoundButtonConfiguration()
'VBA667
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
  .PicDeactTransparent = RGB(255, 0, 0)
  .PicDeactUseTransColor = True
End With
End Sub

See also

PicDeactTransparent Property (Page 2366)
RoundButton Object (Page 2052)

PicDownReferenced Property

Description

TRUE if the picture assigned to the "On" status is to be saved in the RoundButton object. Otherwise, only the associated object reference is saved. BOOLEAN write-read access.
Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the picture assigned to the "On" status will be referenced:

```
Sub RoundButtonConfiguration()
    'VBA668
    Dim objRoundButton As HMIRoundButton
    Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
    With objRoundButton
        .PicDownReferenced = False
    End With
End Sub
```

See also

RoundButton Object (Page 2052)

PicDownTransparent Property

Description

Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "On" status should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "PicDownUseTransColor" property is "True".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Yellow" assigned in the Bitmap object is to be displayed transparent when in the "Deactivated" status.

Sub RoundButtonConfiguration()
'VBA669
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
  .PicDownTransparent = RGB(255, 255, 0)
  .PicDownUseTransColor = True
End With
End Sub

See also

PicDownUseTransColor Property (Page 2369)
RoundButton Object (Page 2052)

PicDownUseTransColor Property

Description

TRUE, when the transparent color defined by the "PicDownTransparent" property for the "On" status should be used. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Yellow" assigned in the Bitmap object is to be displayed transparent when in the "Deactivated" status:

Sub RoundButtonConfiguration()
'VBA670
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
  .PicDownTransparent = RGB(255, 255, 0)
  .PicDownUseTransColor = True
End With
End Sub
PicReferenced Property

Description

TRUE if the picture assigned to the GraphicObject object is to be referenced and not saved in the object. BOOLEAN write-read access.

Example:

The "GraphicObjectConfiguration()" procedure accesses the properties of the graphics object. In this example the assigned picture will be referenced:

```
Sub GraphicObjectConfiguration()
    'VBA671
    Dim objGraphicObject As HMIGraphicObject
    Set objGraphicObject = ActiveDocument.HMIObjects.AddHMIObject("GraphicObject1", "HMIGraphicObject")
    With objGraphicObject
        .PicReferenced = True
    End With
End Sub
```

See also

GraphicObject Object (Page 1975)

PictAlignment property

Description

As the "Picture alignment" attribute, it defines the position and scaling of the picture placed on the button or round button.

- centered: The picture is positioned, centered in the original proportions.
- Left justified: The picture is positioned with original proportions, with left justification on the left side of the button.
- Right justified: The picture is positioned with original proportions, with right justification on the right side of the button.
- Stretched: The picture is scaled to a square and is adapted to the size of the button.
PicTransColor Property

Description

Defines or returns which color of the assigned bitmap object (.bmp, .dib) should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "PicUseTransColor" property is "True".

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "GraphicObjectConfiguration()" procedure accesses the properties of the graphics object. In this example the color "Blue" assigned in the Bitmap object is to be displayed transparent:

Sub GraphicObjectConfiguration()
  'VBA672
  Dim objGraphicObject As HMIGraphicObject
  Set objGraphicObject = ActiveDocument.HMIObjects.AddHMIObject("GraphicObject1", "HMIGraphicObject")
  With objGraphicObject
    .PicTransColor = 16711680
    .PicUseTransColor = True
  End With
End Sub

See also

GraphicObject Object (Page 1975)

PictureDeactivated Property

Description

Defines the picture to be displayed in the "Disable" status or returns the picture name.

The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.
Example:

The "ButtonConfiguration()" procedure accesses the properties of the round button. In this example the pictures for the "On" and "Off" states will be defined:

Sub ButtonConfiguration()
  ' VBA673
  Dim objRoundButton As HMIRoundButton
  Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
  With objRoundButton
    ' To use this example copy a Bitmap-Graphic
    ' to the "GraCS"-Folder of the actual project.
    ' Replace the picturename "TestPicture1.BMP" with the name of
    ' the picture you copied
    .PictureDeactivated = "TestPicture1.BMP"
  End With
End Sub

See also

  RoundButton Object (Page 2052)
  PicReferenced Property (Page 2370)

PictureDown Property

Description

Defines the picture to be displayed in the "On" status or returns the picture name.

The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.
Example:

The "ButtonConfiguration()" procedure accesses the properties of the round button. In this example the pictures for the "On" and "Off" states will be defined:

Sub ButtonConfiguration()
'VBA674
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIOObjects.AddHMIObject("Button1", "HMIButton")
With objButton

' To use this example copy two Bitmap-Graphics
to the "GraCS"-Folder of the actual project.
'Replace the picturenames "TestPicture1.BMP" and "TestPicture2.BMP"
'with the names of the pictures you copied
.PictureDown = "TestPicture1.BMP"
.PictureUp = "TestPicture2.BMP"
End With
End Sub

See also

RoundButton Object (Page 2052)

PictureName Property

Description

Defines the picture to be displayed in the picture window in Runtime or returns the picture name.

The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

```vba
Sub PictureWindowConfig()
  'VBA675
  Dim objPicWindow As HMIPictureWindow
  Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
  With objPicWindow
    .AdaptPicture = False
    .AdaptSize = False
    .Caption = True
    .CaptionText = "Picturewindow in runtime"
    .OffsetLeft = 5
    .OffsetTop = 10
    'Replace the picturename "Test.PDL" with the name of
    'an existing document from your "GraCS"-Folder of your active project
    .PictureName = "Test.PDL"
    .ScrollBars = True
    .ServerPrefix = ""
    .TagPrefix = "Struct."
    .UpdateCycle = 5
    .Zoom = 100
  End With
End Sub
```

See also

PictureWindow Object (Page 2029)

PictureUp Property

Description

Defines the picture to be displayed in the "Off" status or returns the picture name.

The picture (*.BMP or *.DIB) must be located in the "GraCS" directory of the current project so that it can be integrated.
Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the pictures for the "On" and "Off" states will be defined:

```vba
Sub ButtonConfiguration()
    'VBA676
    Dim objButton As HMIButton
    Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
    With objButton
        'To use this example copy two Bitmap-Graphics
to the "GraCS"-Folder of the actual project.
'Replace the picture-names "TestPicture1.BMP" and "TestPicture2.BMP"
'with the names of the pictures you copied
    .PictureDown = "TestPicture1.BMP"
    .PictureUp = "TestPicture2.BMP"
    End With
    End Sub
```

See also

- RoundButton Object (Page 2052)
- Button Object (Page 1921)

PicUpReferenced Property

Description

TRUE if the picture assigned to the "Off" status is to be saved in the RoundButton object. Otherwise, only the associated object reference is saved. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the picture assigned to the "Off" status will be referenced:

```vba
Sub RoundButtonConfiguration()
    'VBA677
    Dim objRoundButton As HMIRoundButton
    Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
    With objRoundButton
        .PicUpReferenced = False
        End With
    End Sub
```
PicUpTransparent Property

Description
Defines or returns which color of the bitmap object (.bmp, .dib) assigned to the "Off" status should be set to "transparent". LONG write-read access.

The color is only set to "Transparent" if the value of the "PicUpUseTransColor" property is "True".

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Blue" assigned in the Bitmap object is to be displayed transparent in the status "Off".

Sub RoundButtonConfiguration()
'VBA678
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObj ect "$Button1", "HMIRoundButton"
With objRoundButton
  .PicUpTransparent = RGB(0, 0, 255)
  .PicUpUseTransColor = True
End With
End Sub

See also
PicUpUseTransColor Property (Page 2377)
RoundButton Object (Page 2052)
PicUpUseTransColor Property

Description

TRUE, when the transparent color defined by the "PicUpTransparent" property for "Off" status should be used. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the color "Blue" assigned in the Bitmap object is to be displayed transparent in the status "Off":

```
Sub RoundButtonConfiguration()
  'VBA679
  Dim objRoundButton As HMIRoundButton
  Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
  With objRoundButton
    .PicUpTransparent = RGB(0, 0, 255)
    .PicUpUseTransColor = True
  End With
End Sub
```

See also

PicUpTransparent Property (Page 2376)
RoundButton Object (Page 2052)

PicUseTransColor Property

Description

TRUE if the transparent color defined with the "PicTransColor" property is to be used for the "Deactivated" status. BOOLEAN write-read access.
Example:
The "GraphicObjectConfiguration()" procedure accesses the properties of the graphics object.
In this example the color "Blue" assigned in the Bitmap object is to be displayed transparent:

Sub GraphicObjectConfiguration()
'VBA680
Dim objGraphicObject As HMIGraphicObject
Set objGraphicObject = ActiveDocument.HMIObjets.AddHMIObject("GraphicObject1",
"HMIGraphicObject")
With objGraphicObject
  .PicTransColor = RGB(0, 0, 255)
  .PicUseTransColor = True
End With
End Sub

See also
PicTransColor Property (Page 2371)
GraphicObject Object (Page 1975)

Pinnable property

Description
Only used internally.

See also
PictureWindow Object (Page 2029)

Pinned property

Description
Only used internally.

See also
PictureWindow Object (Page 2029)
PointCount Property

Description
Defines or returns the number of corner points in the case of the Polygon and Polyline objects. Each corner point has position coordinates and is identified via an index.

Example:
For this example to work, insert a polyline called "Polyline1" into the active picture: The "PolyLineCoordsOutput" procedure then outputs the coordinates of all the corner points in the polyline:

```vba
Sub PolyLineCoordsOutput()
    'VBA681
    Dim iPcIndex As Integer
    Dim iPosX As Integer
    Dim iPosY As Integer
    Dim iIndex As Integer
    Dim objPolyLine As HMIPolyLine
    Set objPolyLine = Application.ActiveDocument.HMIObjects.AddHMIObject("PolyLine1", "HMIPolyLine")

    'Determine number of corners from "PolyLine1":
    iPcIndex = objPolyLine.PointCount

    'Output of x/y-coordinates from every corner:
    For iIndex = 1 To iPcIndex
        With objPolyLine
            .index = iIndex
            iPosX = .ActualPointLeft
            iPosY = .ActualPointTop
        End With
        MsgBox iIndex & ". corner:" & vbCrLf & "x-coordinate: " & iPosX & vbCrLf & "y-coordinate: " & iPosY
    Next iIndex
End Sub
```

List of links
See also
Index Property (Page 2276)
ActualPointTop Property (Page 2110)
ActualPointLeft Property (Page 2109)
PolyLine Object (Page 2038)
Polygon Object (Page 2035)
Position Property

Description
The value for position determines the sequence, in which menu entries and icons are assigned in user-defined menus and toolbars or how user-defined menus are arranged in the menu bar. Write/Read access.

A value of "1" means position 1 (start).

Example:
In the following example the position of all menu entries in the first user-defined menu in the active picture will be output: So that this example will work, first carry out the example shown under the heading "InsertSubMenu".

Sub ShowPositionOfCustomMenuItems()
    'VBA683
    Dim objMenu As HMIMenu
    Dim iMaxMenuItems As Integer
    Dim iPosition As Integer
    Dim iIndex As Integer
    Set objMenu = ActiveDocument.CustomMenus(1)
iMaxMenuItems = objMenu.MenuItems.Count
For iIndex = 1 To iMaxMenuItems
    iPosition = objMenu.MenuItems(iIndex).Position
    MsgBox "Position of the " & iIndex & ". menuitem: " & iPosition
Next iIndex
End Sub

See also
ToolbarItem Object (Page 2083)
MenuItem Object (Page 2013)
Menu Object (Page 2009)
InsertSubmenu Method (Page 1854)

PositiveValue Property

Description
Defines the value for the dynamic property if the configured tag returns a non-zero value, or returns the value.
Example:

Use the BinaryResultInfo property to return the BinaryResultInfo object. In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and the associated property values will be assigned to both the binary value ranges:

Sub AddDynamicDialogToCircleRadiusTypeBool()
'VBA684
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_C", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'"");
With objDynDialog
  .ResultType = hmiResultTypeBool
  .BinaryResultInfo.NegativeValue = 20
  .BinaryResultInfo.PositiveValue = 40
End With
End Sub

See also

NegativeValue Property (Page 2344)
BinaryResultInfo Object (Page 1918)
VBA Reference (Page 1745)

PredefinedAngels Property

Description

Defines or returns the depth of the display of the 3DBarGraph object. Value range from 0 to 3.

<table>
<thead>
<tr>
<th>Display</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cavalier</td>
<td>0</td>
</tr>
<tr>
<td>Isometric</td>
<td>1</td>
</tr>
<tr>
<td>Axonometric</td>
<td>2</td>
</tr>
<tr>
<td>Freely Defined</td>
<td>3</td>
</tr>
</tbody>
</table>
Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the depth display will be set to "Isometric":

```vba
Sub HMI3DBarGraphConfiguration()
    'VBA685
    Dim obj3DBar As HMI3DBarGraph
    Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3Bar1", "HMI3DBarGraph")
    With obj3DBar
        'Depth-angle a = 15 degrees
        .AngleAlpha = 15
        .PredefinedAngles = 1
        'Depth-angle b = 45 degrees
        .AngleBeta = 45
    End With
End Sub
```

See also

3DBarGraph Object (Page 1894)

Pressed Property

Description

TRUE, when the Button or RoundButton object is pressed. BOOLEAN write-read access.

Example:

The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the RoundButton object will be set to "Pressed":

```vba
Sub RoundButtonConfiguration()
    'VBA686
    Dim objRoundButton As HMIRoundButton
    Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
    With objRoundButton
        .Pressed = True
    End With
End Sub
```

See also

RoundButton Object (Page 2052)
PrioAlarm..Warning property

Description

Specifies the priority at one of the following states or message types:

- Alarm
- Warning
- Tolerance
- AS Process Control Error
- AS Control System Fault
- Operator request

PrioBit16..31 property

Description

The property indicates the priority of the respective bit in the group value for the alarm evaluation for the advanced analog and status display. The alarm evaluation starts at the highest priority (priority 1). Bits that are not used for the alarm evaluation are assigned priority 0.

If the group value contains multiple bits, the priority determines which status is displayed.

Process Property

Description

Defines or returns presetting for the value to be displayed.

This value is used in Runtime when the associated tag cannot be connected or updated when a picture is started.
Example:
The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the default value will be set to "100":

Sub HMI3DBarGraphConfiguration()
  'VBA687
  Dim obj3DBar As HMI3DBarGraph
  Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
  With obj3DBar
    'Depth-angle a = 15 degrees
    .AngleAlpha = 15
    'Depth-angle b = 45 degrees
    .AngleBeta = 45
    .Process = 100
  End With
End Sub

See also
Slider object (Page 2062)
OptionGroup Object (Page 2025)
CheckBox Object (Page 1924)
BarGraph Object (Page 1914)
3DBarGraph Object (Page 1894)

Process property

Description
Here the first tag is stored that is used for status value calculation for the "HMIAdvancedStateDisplay" object.

Use the "BitPosition0..3" properties to specify the bit position of these tags that is taken into account for the status value calculation. This results in the statuses to which you can then assign pictures.

Process1 property

Description
Here the second tag is stored that is used for status value calculation for the "HMIAdvancedStateDisplay" object.
Use the "BitPosition0..3" properties to specify the bit position of these tags that is taken into account for the status value calculation. This results in the statuses to which you can then assign pictures.

**Process2 property**

**Description**

Here the third tag is stored that is used for status value calculation for the "HMIAdvancedStateDisplay" object.

Use the "BitPosition0..3" properties to specify the bit position of these tags that is taken into account for the status value calculation. This results in the statuses to which you can then assign pictures.

**Process3 property**

**Description**

Here the fourth tag is stored that is used for status value calculation for the "HMIAdvancedStateDisplay" object.

Use the "BitPosition0..3" properties to specify the bit position of these tags that is taken into account for the status value calculation. This results in the statuses to which you can then assign pictures.

**ProfileName Property**

**Description**

Returns the name of the specified application. Read only access.

**Example:**

In this example the name of the "Graphics Designer" application will be output:

```vba
Sub ShowProfileName()
'VBA688
MsgBox ApplicationProfileName
End Sub
```

**See also**

Application Object (Page 1909)
ProgID Property

Description

Returns the ProgID of an ActiveX Control. STRING read access.

Example:

In the following example the ActiveX Control "WinCC Gauge Control" is inserted in the active picture. The ProgID is then output:

```vba
Sub AddActiveXControl()
    'VBA689
    Dim objActiveXControl As HMIActiveXControl
    Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge", "XGAUGE.XGaugeCtrl.1")
    With ActiveDocument
        .HMIObjects("WinCC_Gauge").Top = 40
        .HMIObjects("WinCC_Gauge").Left = 40
        MsgBox "ProgID of ActiveX-control: " & .HMIObjects("WinCC_Gauge").ProgID
    End With
End Sub
```

See also

ActiveXControl Object (Page 1900)

AddActiveXControl Method (Page 1795)

ProjectName Property

Description

Returns the project name. Read access.
Example:

In this example the name and type of the loaded project will be output.

Sub ShowProjectInfo()
  'VBA690
  Dim iProjectType As Integer
  Dim strProjectName As String
  Dim strProjectType As String
  iProjectType = Application.ProjectType
  strProjectName = Application.ProjectName
  Select Case iProjectType
    Case 0
      strProjectType = "Single-User System"
    Case 1
      strProjectType = "Multi-User System"
    Case 2
      strProjectType = "Client System"
  End Select
  MsgBox "Projecttype: " & strProjectType & vbCrLf & "Projectname: " & strProjectName
End Sub

See also

Application Object (Page 1909)

ProjectType Property

Description

Returns the project type. Value range from 0 to 2. Read access.

<table>
<thead>
<tr>
<th>Project type</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-user project</td>
<td>0</td>
</tr>
<tr>
<td>Multi-user project</td>
<td>1</td>
</tr>
<tr>
<td>client project</td>
<td>2</td>
</tr>
</tbody>
</table>
Example:

In this example the name and type of the loaded project will be output:

```
Sub ShowProjectInfo()
'VBA691
Dim iProjectType As Integer
Dim strProjectName As String
Dim strProjectType As String
iProjectType = Application.ProjectType
strProjectName = Application.ProjectName
Select Case iProjectType
Case 0
strProjectType = "Single-User System"
Case 1
strProjectType = "Multi-User System"
Case 2
strProjectType = "Client System"
End Select
MsgBox "Projecttype: " & strProjectType & vbCrLf & "Projectname: " & strProjectName
End Sub
```

See also

Application Object (Page 1909)

Properties Property

Description

Returns a Properties listing containing all the properties of the specified object. Read only access.

To return an element from the Properties listing you can use either the index number or the name of the VBA property.

You must use the Properties property if, for example, you wish to access the properties of objects located in a group object.

Example:

Examples showing how to use the Properties property can be found in this documentation under the following headings:

- "Editing Objects with VBA"
- "Group objects"
- "Customized Objects"
See also

HMIObject Object (Page 1988)
Customized Objects (Page 1682)
Group Objects (Page 1674)
Editing Objects with VBA (Page 1660)

Prototype Property

Description

Returns the function heading of a script. The function heading is assigned by default if no source code is configured.

Example:

In the following example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle will enlarge every time you click the button. In this case only the prototype of the VB script is output:

```vba
Sub ExampleForPrototype()
'VBA692
Dim objButton As HMIButton
Dim objCircleA As HMICircle
Dim objEvent As HMIEvent
Dim objVBScript As HMIScriptInfo
Set objCircleA = ActiveDocument.HMIObjects.AddHMIObject("CircleA", "HMICircle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objCircleA
  .Top = 100
  .Left = 100
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 200
  .Text = "Increase Radius"
End With
'On every mouseclick the radius have to increase:
Set objEvent = objButton.Events(1)
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
MsgBox objVBScript.Prototype
End Sub
```

See also

ScriptInfo Object (Page 2058)
**QualityCodeStateChecked Properties**

**Description**

TRUE, if the quality code of the specified tag is used in Dynamic dialog for dynamization. BOOLEAN write-read access.
Example:

In the following example the radius of a circle is given dynamics with the Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA816
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
                                     "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .Activate analysis of quality codestate
    .QualityCodeStateChecked = True
  End With
  With objDynDialog.QualityCodeStateValues(1)
    'define a value for every state:
    .VALUE_BAD_COMMLUV = 20
    .VALUE_BAD_COMMNUV = 30
    .VALUE_BAD_CONFERROR = 40
    .VALUE_BAD_DEVICE = 60
    .VALUE_BAD_MISCSTATES = 70
    .VALUE_BAD_NONSPECIFIC = 80
    .VALUE_BAD_NOTCONNECTED = 90
    .VALUE_BAD_OUTOFSEFR = 100
    .VALUE_BAD_PROCRELNOM = 110
    .VALUE_BAD_PROCRELSUB = 120
    .VALUE_HIGHLIMITED = 130
    .VALUE_LOWLIMITED = 140
    .VALUE_UNCERT_ENGVHIGHLIM = 150
    .VALUE_UNCERT_ENGVLOWLIM = 160
    .VALUE_UNCERT_INITVAL = 170
    .VALUE_UNCERT_LUV = 180
    .VALUE_UNCERT_MAINTDEM = 190
    .VALUE_UNCERT_MISCSTATES = 200
    .VALUE_UNCERT_NONSPECIFIC = 210
    .VALUE_UNCERT_PROCRELNOM = 220
    .VALUE_UNCERT_SIMVAL = 230
    .VALUE_UNCERT_SUBSTSET = 240
  End With
End Sub
```

See also

DynamicDialog Object (Page 1955)
QualityCodeStateValues Property

Description

Returns the QualityCodeStateValues listing. Use the QualityCodeStateValues property with the Item property to assign a value to the quality code status to be used for dynamization.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic
dialog Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a
quality code, a substitute value (ElseCase property) is defined:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA817
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMIcircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMIcircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  ,'Activate analysis of qualitycodestate
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  ,'define a value for every state:
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSMSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSEVR = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub

See also

DynamicDialog Object (Page 1955)
QualityCodeStateValues Object (Listing) (Page 2046)
Radius Property

Description

Defines or returns the radius in the case of the following objects:

- Circle: Radius in pixels (0 to 10000)
- CircularArc: Radius in pixels (0 to 10000)
- PieSegment: Radius in pixels (0 to 10000)
- RoundButton: Radius in pixels (0 to 10000)

Example:

The "PieSegmentConfiguration()" procedure accesses the properties of the Pie Segment. In this example the radius will be set to "80":

Sub PieSegmentConfiguration()
  'VBA693
  Dim objPieSegment As HMIPieSegment
  Set objPieSegment = ActiveDocument.HMIObjects.AddHMIObject("PieSegment1", "HMIPieSegment")
  With objPieSegment
    .StartAngle = 40
    .EndAngle = 180
    .Radius = 80
  End With
End Sub

See also

RoundButton Object (Page 2052)
PieSegment Object (Page 2032)
CircularArc Object (Page 1931)
Circle Object (Page 1928)

RadiusHeight Property

Description

Defines or returns the vertical radius in pixels (0 to 5000) in the case of elliptical objects (Ellipse, EllipseArc, EllipseSegment).
Example:

The "EllipseConfiguration()" procedure accesses the properties of the ellipse object. In this example the horizontal radius will be set to "60":

```vba
Sub EllipseConfiguration()
  'VBA694
  Dim objEllipse As HMIEllipse
  Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("Ellipse1", "HMIEllipse")
  With objEllipse
    .RadiusHeight = 60
    .RadiusWidth = 40
  End With
End Sub
```

See also

- RadiusWidth Property (Page 2395)
- EllipseSegment Object (Page 1963)
- EllipseArc Object (Page 1960)
- Ellipse Object (Page 1957)

RadiusWidth Property

Description

Defines or returns the horizontal radius in pixels (0 to 5000) in the case of elliptical objects (Ellipse, EllipseArc, EllipseSegment).

Example:

The "EllipseConfiguration()" procedure accesses the properties of the ellipse object. In this example the horizontal radius will be set to "40":

```vba
Sub EllipseConfiguration()
  'VBA695
  Dim objEllipse As HMIEllipse
  Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("Ellipse1", "HMIEllipse")
  With objEllipse
    .RadiusHeight = 60
    .RadiusWidth = 40
  End With
End Sub
```
See also
- RadiusHeight Property (Page 2394)
- EllipseSegment Object (Page 1963)
- EllipseArc Object (Page 1960)
- Ellipse Object (Page 1957)

RangeTo Property

Description
Defines or returns the analog value range.

Example:
An example showing how to use the RangeTo property can be found in this documentation under the heading "AnalogResultInfos Object (Listing)".

See also
- Value Property (Page 2469)
- AnalogResultInfos Object (Listing) (Page 1908)
- AnalogResultInfo Object (Page 1907)

ReferenceRotationLeft Property

Description
Defines or returns the X-coordinate of the reference point about which the object should be rotated in Runtime.

The value of the X-coordinate is relative to the object width. Enter the value in percent starting from the left edge of the rectangle enclosing the object.
Example:

The "PolyLineConfiguration()" procedure accesses the properties of the PolyLine object. In this example, the coordinates of the reference point will be set to 50% of the object width and 50% of the object height:

```vba
Sub PolyLineConfiguration()
    'VBA696
    Dim objPolyLine As HMIPolyLine
    Set objPolyLine = ActiveDocument.HMIObjects.AddHMIObject("PolyLine1", "HMIPolyLine")
    With objPolyLine
        .ReferenceRotationLeft = 50
        .ReferenceRotationTop = 50
    End With
End Sub
```

See also

- RotationAngle Property (Page 2400)
- ReferenceRotationTop Property (Page 2397)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- Line Object (Page 2004)

ReferenceRotationTop Property

Description

Defines or returns the Y-coordinate of the reference point about which the object should be rotated in Runtime.

The value of the Y-coordinate is relative to the object width. Enter the value in percent starting from the top edge of the rectangle enclosing the object.
Example:

The "PolyLineConfiguration()" procedure accesses the properties of the PolyLine object. In this example, the coordinates of the reference point will be set to 50% of the object width and 50% of the object height:

```vba
Sub PolyLineConfiguration()
'VBA697
Dim objPolyLine As HMIPolyLine
Set objPolyLine = ActiveDocument.HMIObjects.AddHMIObject("PolyLine1", "HMIPolyLine")
With objPolyLine
  .ReferenceRotationLeft = 50
  .ReferenceRotationTop = 50
End With
End Sub
```

See also

- RotationAngle Property (Page 2400)
- ReferenceRotationLeft Property (Page 2396)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- Line Object (Page 2004)

Relevant Property

Description

TRUE when the "GroupDisplay", "AdvancedAnalogDisplay" or "AdvancedStateDisplay" object is taken into account when forming the group display. BOOLEAN write-read access.

Example

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the object for forming the group display will be considered:

```vba
Sub GroupDisplayConfiguration()
'VBA698
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .Relevant = True
End With
End Sub
```
See also

Group Object (Page 1978)

**ResultType Property**

**Description**

Defines or returns the value range evaluation type in the Dynamic dialog.

**Example:**

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog, a tag name will be assigned and the associated property values will be assigned to both the binary value ranges:

```vba
Sub AddDynamicDialogToCircleRadiusTypeBinary()
    'VBA699
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_C", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeBool
        .BinaryResultInfo.NegativeValue = 20
        .BinaryResultInfo.PositiveValue = 40
    End With
End Sub
```

See also

DynamicDialog Object (Page 1955)

**RightComma Property**

**Description**

Defines or returns the number of decimal places (0 to 20) for the BarGraph object.
Example:

The "BarGraphConfiguration()" procedure configures In this example the number of decimal places will be limited to 4.

Sub BarGraphConfiguration()
'VBA700
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
 .RightComma = 4
End With
End Sub

See also

BarGraph Object (Page 1914)

RotationAngle Property

Description

Line, Polygon and PolyLine
Defines or returns the rotation angle of the following objects in degrees: Line, Polygon, PolyLine.
The object is displayed in Runtime only rotated clockwise around the reference point by the specified value (starting from the configured starting position).

T-piece
Defines or returns the orientation of a T-piece in degrees. The attribute can only assume one of four values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The standard position of the T-piece is the shape of the letter &quot;T&quot;</td>
</tr>
<tr>
<td>90</td>
<td>The &quot;leg&quot; of the &quot;T&quot; points towards the left</td>
</tr>
<tr>
<td>180</td>
<td>The &quot;leg&quot; of the &quot;T&quot; points upwards</td>
</tr>
<tr>
<td>270</td>
<td>The &quot;leg&quot; of the &quot;T&quot; points to the right</td>
</tr>
</tbody>
</table>

Other values are automatically converted to modulus 360 and rounded up or down to the nearest permissible value.
The T-piece is shown rotated around the center point in the project and in Runtime.
Example:

The "PolyLineConfiguration()" procedure accesses the properties of the PolyLine object. In this example the object will be rotated by 45° in Runtime:

```vba
Sub PolyLineConfiguration()
  'VBA701
  Dim objPolyLine As HMIPolyLine
  Set objPolyLine = ActiveDocument.HMIObjects.AddHMIObject("PolyLine1", "HMIPolyLine")
  With objPolyLine
    .ReferenceRotationLeft = 50
    .ReferenceRotationTop = 50
    .RotationAngle = 45
  End With
End Sub
```

See also

- ReferenceRotationTop Property (Page 2397)
- ReferenceRotationLeft Property (Page 2396)
- PolyLine Object (Page 2038)
- Polygon Object (Page 2035)
- Line Object (Page 2004)

RoundCornerHeight Property

Description

Defines or returns the corner radius of the RoundRectangle object.

Enter the value as a percentage of half the height of the object.
Example:

The "RoundRectangleConfiguration()" procedure accesses the properties of the object RoundRectangle. In this example the corner radius will be set to 25% (height) and 50% (width).

Sub RoundRectangleConfiguration()
  Dim objRoundRectangle As HMIRoundRectangle
  Set objRoundRectangle = ActiveDocument.HMIObjects.AddHMIObject("RoundRectangle1", "HMIRoundRectangle")
  With objRoundRectangle
    .RoundCornerHeight = 25
    .RoundCornerWidth = 50
  End With
End Sub

See also

RoundCornerWidth Property (Page 2402)
RoundRectangle Object (Page 2055)

RoundCornerWidth Property

Description

Defines or returns the corner radius of the RoundRectangle object.

Enter the value as a percentage of half the width of the object.

Example:

The "RoundRectangleConfiguration()" procedure accesses the properties of the object RoundRectangle. In this example the corner radius will be set to 25% (height) and 50% (width):

Sub RoundRectangleConfiguration()
  Dim objRoundRectangle As HMIRoundRectangle
  Set objRoundRectangle = ActiveDocument.HMIObjects.AddHMIObject("RoundRectangle1", "HMIRoundRectangle")
  With objRoundRectangle
    .RoundCornerHeight = 25
    .RoundCornerWidth = 50
  End With
End Sub
See also

RoundCornerHeight Property (Page 2401)
RoundRectangle Object (Page 2055)

S

SameSize Property

Description

TRUE, when all four buttons of a Group Display object have the same size. BOOLEAN write-read access.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example all four buttons will have the same size.

Sub GroupDisplayConfiguration()
'VBA704
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .SameSize = True
End With
End Sub

See also

GroupDisplay Object (Page 1980)

ScaleColor Property

Description

Defines or returns the color of the scale. LONG write-read access.
The "Scaling" property must be set to TRUE for the color to be displayed.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).
Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "BarGraphConfiguration()" procedure configures In this example the scale will be displayed and the scale color will be set to "Red":

```vba
Sub BarGraphConfiguration()
  'VBA705
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    .Scaling = True
    .ScaleColor = RGB(255, 0, 0)
  End With
End Sub
```

See also
Scaling Property (Page 2405)
BarGraph Object (Page 1914)

ScaleTicks Property

Description
Defines or returns the number of scale sections for the BarGraph object.
A scale section is a part of the scale bounded by two long scale strokes or division ticks. If you assign a value of "0" to the property, the appropriate scale marks will be calculated automatically.

Example:
The "BarGraphConfiguration()" procedure configures In this example the number of scale sections will be set to "10".

```vba
Sub BarGraphConfiguration()
  'VBA706
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    .Scaling = True
    .ScaleTicks = 10
  End With
End Sub
```
See also

BarGraph Object (Page 1914)

Scaling Property

Description

TRUE if a scale is also used to display the values in the case of the BarGraph object. BOOLEAN
write-read access.

Example:

The "BarGraphConfiguration()" procedure configures the properties of the BarGraph object.
In this example the scale will be displayed and the scale color will be set to "Red":

```
Sub BarGraphConfiguration()
'VBA707
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjecnts.AddHMIObject("Bar1", "HMI BarGraph")
With objBarGraph
  .Scaling = True
  .ScaleColor = RGB(255, 0, 0)
End With
End Sub
```

See also

BarGraph Object (Page 1914)

ScalingMode property

Description

Defines the size to display the objects of the faceplate instance.

Default Like scaling mode "proportional"
1 : 1 The faceplate type is displayed in the original size in the faceplate instance. If
the faceplate instance is too small, the size of the faceplate instance is adapted
to the size of the faceplate type.
Proportional The faceplate type is scaled in proportion with the size of the faceplate instance.
Example

ScalingType Property

Description

Defines or returns the type of bar scaling. Value range from 0 to 2.

The "Scaling" property must be set to TRUE for the color to be displayed.

<table>
<thead>
<tr>
<th>Bar Scaling</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>1</td>
</tr>
<tr>
<td>Automatic</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:

The "BarGraphConfiguration()" procedure configures In this example the bar scaling will be set to "Linear":

```vba
Sub BarGraphConfiguration()
  'VBA708
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjets.AddHMIObjet("Bar1", "HMIBarGraph")
  With objBarGraph
    .ScalingType = 0
    .Scaling = True
  End With
End Sub
```

See also

Scaling Property (Page 2405)
BarGraph Object (Page 1914)
ScriptType Property

Description

Returns the script type (C or VBS) which was used to make a property or event dynamic. Read only access.

Example:

In the following example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle will enlarge every time you click the button. In this case the script type will be output:

```vba
Sub ExampleForPrototype()
'VBA709
Dim objButton As HMIButton
Dim objCircleA As HMICircle
Dim objEvent As HMIEvent
Dim objVBScript As HMIScriptInfo
Dim strScriptType As String
Set objCircleA = ActiveDocument.HMIObs.AddHMIObject("CircleA", "HMICircle")
Set objButton = ActiveDocument.HMIObs.AddHMIObject("myButton", "HMIButton")
With objCircleA
  .Top = 100
  .Left = 100
End With
With objButton
  .Top = 10
  .Left = 10
  .Width = 200
  .Text = "Increase Radius"
End With
'On every mouseclick the radius have to increase:
Set objEvent = objButton.Events(1)
Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
Select Case objVBScript.ScriptType
  Case 0
    strScriptType = "VB script is used"
  Case 1
    strScriptType = "C-Skript is used"
End Select
MsgBox strScriptType
End Sub
```

See also

ScriptInfo Object (Page 2058)
ScrollBars Property

Description

TRUE if the picture window has scroll bars in Runtime. BOOLEAN write-read access.

Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

```vba
Sub PictureWindowConfig()
'VBA710
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10
  'Replace the picturename "Test.PDL" with the name of
  'an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ServerPrefix = ""
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub
```

See also

PictureWindow Object (Page 2029)

ScrollPositionX Property

Description

Specifies the horizontal positioning of the scroll bar in a picture window with slider, or returns its value.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

Sub PictureWindowConfig()
'VBA808
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10
  'Replace the picturename "Test.PDL" with the name of
  'an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ScrollPositionX = 50
  .ScrollPositionY = 50
  .ServerPrefix = ""
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub

See also

PictureWindow Object (Page 2029)

ScrollPositionY Property

Description

Specifies the vertical positioning of the scroll bar in a picture window with slider, or returns its value.
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

Sub PictureWindowConfig()
'VBA809
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObj ect.s.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10
  'Replace the picturename "Test.PDL" with the name of an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ScrollPositionX = 50
  .ScrollPositionY = 50
  .ServerPrefix = ""
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub

See also

PictureWindow Object (Page 2029)

ScrollPosX Property

Description

Defines or returns the X position of the scroll bars for the View object.
Example:

In the following example a copy of the active picture is created and then activated. The position of the scroll bars will be set to 40 (X) and 10 (Y):

Sub CreateViewAndActivateView()
  Dim objView As HMIView
  Set objView = ActiveDocument.Views.Add
  objView.Activate
  objView.ScrollPosX = 40
  objView.ScrollPosY = 10
End Sub

See also
- ScrollPosY Property (Page 2411)
- View Object (Page 2102)

ScrollPosY Property

Description

Defines or returns the Y position of the scroll bars for the View object.

Example:

In the following example a copy of the active picture is created and then activated. The position of the scroll bars will be set to 40 (X) and 10 (Y):

Sub CreateViewAndActivateView()
  Dim objView As HMIView
  Set objView = ActiveDocument.Views.Add
  objView.Activate
  objView.ScrollPosX = 40
  objView.ScrollPosY = 10
End Sub

See also
- ScrollPosX Property (Page 2410)
- View Object (Page 2102)
SelBGColor Property

Description

Defines or returns the background color for the selected entry in the case of the TextList object. LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the background color for the selected entry will be set to "Red":

```vba
Sub TextListConfiguration()
    Dim objTextList As HMITextList
    'Neue TextListe ins aktuelle Bild einfügen:
    Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
    With objTextList
        .SelBGColor = RGB(255, 0, 0)
    End With
End Sub
```

See also

TextList Object (Page 2076)

Selected Property

Description

TRUE if an object is selected in the picture. BOOLEAN write-read access.
Example:

In the following example two new objects will be inserted in the active picture and then selected:

```vba
Sub SelectObjects()
  'VBA714
  Dim objCircle As HMICircle
  Dim objRectangle As HMIRectangle
  Dim objGroup As HMIGroup
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
  With objCircle
    .Top = 40
    .Left = 40
    .Selected = True
  End With
  With objRectangle
    .Top = 80
    .Left = 80
    .Selected = True
  End With
  MsgBox "Objects selected!"
End Sub
```

See also

HMIOBJECT Object (Page 1988)

Selection Property

Description

Returns a listing containing all the objects selected in the specified picture.

To return an element from the Selection listing you can use either the index number or the object name.

You can use the Selection property, for example, to select all the objects in the picture.

Example:

In the following example all the objects in the active picture are selected:

```vba
Sub SelectAllObjectsInActiveDocument()
  'VBA715
  ActiveDocument.Selection.SelectAll
End Sub
```
See also

SelectedObjects object (Listing) (Page 2060)
Document Object (Page 1949)

SelIndex property

Description

Defines or returns the index of which the associated text is highlighted in the combobox or list box.

SelText property

Description

Shows the text defined with the "SelIndex" property which is highlighted in the ComboBox or ListBox object. You cannot directly change the "Selected text" attribute. You change the "Selected text" attribute by changing the "Selected box" attribute or the text itself in the "Font" properties group.

SelTextColor Property

Description

Defines or returns the text color for the selected entry in the TextList object. LONG write-read access.

**Determination of Color Value**

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the text color for the selected entry will be set to "Yellow":

```vba
Sub TextListConfiguration()
'VBA716
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
  .SelTextColor = RGB(255, 255, 0)
End With
End Sub
```

See also

TextList Object (Page 2076)

ServerName Property

Description

Returns the name of the specified ActiveX Control or of the embedded object. Read only access.

Example

In the following example the ActiveX Control "WinCC Gauge Control" will be inserted in the active picture and the name of the ActiveX Control will be output:

```vba
Sub AddActiveXControl()
'VBA717
Dim objActiveXControl As HMIActiveXControl
Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge", "XGAUGE.XGaugeCtrl.1")
With objActiveXControl
  .Top = 40
  .Left = 60
  MsgBox .Properties("ServerName").value
End With
End Sub
```

See also

ActiveXControl Object (Page 1900)
ServerPrefix Property

Description

Defines the server which will hold the picture that is displayed in the picture window in Runtime, or returns the name of the server.

Enter the server name followed by two colons: "<Servername>::". No check is made as to whether the server actually exists.

Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

```vba
Sub PictureWindowConfig()
  'VBA718
  Dim objPicWindow As HMIPictureWindow
  Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
  With objPicWindow
    .AdaptPicture = False
    .AdaptSize = False
    .Caption = True
    .CaptionText = "Picturewindow in runtime"
    .OffsetLeft = 5
    .OffsetTop = 10
    'Replace the picturename "Test.PDL" with the name of an existing document from your "GraCS"-Folder of your active project
    .PictureName = "Test.PDL"
    .ScrollBars = True
    .ServerPrefix = "my_Server::"
    .TagPrefix = "Struct."
    .UpdateCycle = 5
    .Zoom = 100
  End With
End Sub
```

See also

PictureWindow Object (Page 2029)

Shortcut Property

Description

Defines or returns a shortcut key sequence for a user-defined menu entry or user-defined icon.
The following keys are permitted in combination with <CTRL>, <ALT> and <SHIFT>:

- Function keys <F1> to <F12>
- The letter keys <A> to <Z> and the number keys <0> to <9>.

The following are not supported: the keys on the alphanumeric keypad, the cursor keys (e.g. <Page Up>) and the remaining function keys such as <RETURN> and <ESC>. No distinction is made upper and lower case. Key combinations with two or more letters or numbers are not permitted, such as "CTRL+A+B", but the combination with two additional keys such as <CTRL+ALT+A" is allowed.

**Notes on using the ShortCut property**

The key sequences used must be unique within the user-defined menus and toolbars in a picture. Key sequences that you configure with VBA have priority over any key sequences that may be present in the Graphics Designer. Within the user-defined menus and toolbars, picture-specific key sequences have priority over application-specific key sequences.

**Note**

Shortcut key sequences are only executed if the menu entry or the icon is visible and active.
Example:

In the following example, a user-defined menu with two menus entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line. The first menu entry receives the shortcut key sequence <CTRL+SHIFT+M> for retrieval:

```
Sub CreateDocumentMenus()
  'VBA719
  Dim objDocMenu As HMIMenu
  Dim objMenuItem As HMIMenuItem
  Dim objSubMenu As HMIMenuItem
  
  'Add menu to menubar:
  Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
  
  'Add menuitems to the new menu:
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "&My first Menuitem")
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second Menuitem")
  
  'Add seperator to menu:
  Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")
  
  'Add submenu to the menu:
  Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "My first submenu")
  
  'Add menuitems to the submenu:
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "My first submenuitem")
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "My second submenuitem")
  
  ActiveDocument.CustomMenus("DocMenu1").MenuItems(1).ShortCut = "CTRL+SHIFT+M"
End Sub
```

See also

- Configuring Menus and Toolbars (Page 1626)
- ToolbarItem Object (Page 2083)
- MenuItem Object (Page 2013)

ShowBadTagState property

Description

Determines if the object is grayed out when a bad quality code or tag status is detected. At both objects, "HMIAdvancedAnalogDisplay" and "HMIAdvancedStateAnalogDisplay", the property is used to specify whether the settings for the "PaintColor_QualityCodeBad" and "PaintColor_QualityCodeUnCertain" properties are used.
SignificantMask Property

Description

Needed in Runtime for displaying the active message class with the highest priority in the GroupDisplay object.

The value of the SignificantMask property represents an internal system output value does not require any specific configuration by the user. Updating takes place in Runtime by clicking on the object.

Example:

```
--
```

See also

GroupDisplay Object (Page 1980)

Simulation property

Description

Specifies the interconnection with any tag that is used for simulation.

SimulationBit property

Description

Shows the bit position of the linked simulation tags that is used for evaluation.

The value of the simulation tag is only evaluated with the alarm status "OK".

Size Property

Description

Defines or returns the font size in points for a language-dependent font.
Example:

The following example sets the font attributes of a button for French and English:

```vba
Sub ExampleForLanguageFonts()
'VBA721
Dim colLangFonts As HMILanguageFonts
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
objButton.Text = "DefText"
Set colLangFonts = objButton.LDFonts

' Set font-properties for french:
With colLangFonts.ItemByLCID(1036)
    .Family = "Courier New"
    .Bold = True
    .Italic = False
    .Underlined = True
    .Size = 12
End With

' Set font-properties for english:
With colLangFonts.ItemByLCID(1033)
    .Family = "Times New Roman"
    .Bold = False
    .Italic = True
    .Underlined = False
    .Size = 14
End With
End Sub
```

See also

- Underlined Property (Page 2460)
- Parent Property (Page 2359)
- LanguageID Property (Page 2289)
- Italic Property (Page 2282)
- Family Property (Page 2229)
- Bold Property (Page 2151)
- Application Property (Page 2121)
- LanguageFont Object (Page 1996)
Sizeable Property

Description

TRUE if the size of the ApplicationWindow and PictureWindow objects can be changed in Runtime. BOOLEAN write-read access.

Example:

The "ApplicationWindowConfig" procedure accesses the properties of the application window. In this example it is intended that the application window can be resized in Runtime:

```vba
Sub ApplicationWindowConfig()
  'VBA722
  Dim objAppWindow As HMIApplicationWindow
  Set objAppWindow = ActiveDocument.HMIObjects.AddHMIObject("AppWindow1", 
    "HMIApplicationWindow")
  With objAppWindow
    .Sizeable = True
  End With
End Sub
```

See also

PictureWindow Object (Page 2029)
ApplicationWindow Object (Page 1911)

SmallChange Property

Description

Defines how many steps the controller can be moved with one mouse click or returns the value.

Example:

The "SliderConfiguration()" procedure accesses the properties of the slider. In this example the number of steps will be set to "4":

```vba
Sub SliderConfiguration()
  'VBA723
  Dim objSlider As HMISlider
  Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
  With objSlider
    .SmallChange = 4
  End With
End Sub
```
See also

Slider object (Page 2062)

SnapToGrid Property

Description

TRUE if objects in the picture are aligned on the grid (which is invisible). BOOLEAN write-read access.

Example:

In the following example, the alignment of objects in the active picture on the grid is activated:

```vba
Sub ActivateSnapToGrid()
  'VBA724
  ActiveDocument.SnapToGrid = True
End Sub
```

See also

Document Object (Page 1949)

SourceLink Property

Description

Returns the Source object. Use the SourceLink property to configure the source object in the case of a direct connection.
Example:

In the following example the X position of "Rectangle_A" is copied to the Y position of "Rectangle_B" in Runtime by clicking on the button:

Sub DirectConnection()
'VBA725
Dim objButton As HMIButton
Dim objRectangleA As HMIRectangle
Dim objRectangleB As HMIRectangle
Dim objEvent As HMIEvent
Dim objDirConnection As HMIDirectConnection
Set objRectangleA = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_A", "HMIRectangle")
Set objRectangleB = ActiveDocument.HMIObjects.AddHMIObject("Rectangle_B", "HMIRectangle")
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
With objRectangleA
 .Top = 100
 .Left = 100
End With
With objRectangleB
 .Top = 250
 .Left = 400
 .BackColor = RGB(255, 0, 0)
End With
With objButton
 .Top = 10
 .Left = 10
 .Width = 100
 .Text = "SetPosition"
End With,
'Directconnection is initiated by mouseclick:
Set objDirConnection = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectConnection)
With objDirConnection
 'Sourceobject: Property "Top" of Rectangle_A
 .SourceLink.Type = hmiSourceTypeProperty
 .SourceLink.ObjectName = "Rectangle_A"
 .SourceLink.AutomationName = "Top"
 ,
 'Targetobject: Property "Left" of Rectangle_B
 .DestinationLink.Type = hmiDestTypeProperty
 .DestinationLink.ObjectName = "Rectangle_B"
 .DestinationLink.AutomationName = "Left"
End With
End Sub

See also

Type Property (Page 2450)
ObjectName Property (Page 2347)
AutomationName Property (Page 2124)
SourceCode Property

Description

Defines or returns the source code of a C script or VB script.

If you assign a C script to the SourceCode property, you must enter only the program code located between the braces ("{
}").

If you assign a VB script to the SourceCode property, you must enter only the program code located between the Sub and EndSub keywords.

Note

If you use single quote marks (') or double quote marks (") in the program code, you must enter an additional quote mark in front of every single or double quote mark so that the program code can be correctly interpreted in the VBA editor.

The Compiled property returns TRUE if the source code was successfully compiled.
Example:
In the following example a button and a circle will be inserted in the active picture. In Runtime the radius of the circle will enlarge every time you click the button. A VB script will be used for this purpose:

```vba
Sub IncreaseCircleRadiusWithVBScript()
  'VBA726
  Dim objButton As HMIButton
  Dim objCircleA As HMICircle
  Dim objEvent As HMIEvent
  Dim objVBScript As HMIScriptInfo
  Dim strCode As String
  strCode = "Dim objCircle" & vbCrLf & "Set objCircle = "
  strCode = strCode & "hmiRuntime.ActiveScreen.ScreenItems(""CircleVB"")"
  strCode = strCode & vbCrLf & "objCircle.Radius = objCircle.Radius + 5"
  Set objCircleA = ActiveDocument.HMIObjects.AddHMIObject("CircleVB", "HMICircle")
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
  With objCircleA
    .Top = 100
    .Left = 100
  End With
  With objButton
    .Top = 10
    .Left = 10
    .Width = 200
    .Text = "Increase Radius"
  End With
  'On every mouseclick the radius have to increase:
  Set objEvent = objButton.Events(1)
  Set objVBScript = objButton.Events(1).Actions.AddAction(hmiActionCreationTypeVBScript)
  objVBScript.SourceCode = strCode
  Select Case objVBScript.Compiled
    Case True
      MsgBox "Compilation ok!"
    Case False
      MsgBox "Error on compilation!"
  End Select
End Sub
```

See also
- Compiled Property (Page 2197)
- ScriptInfo Object (Page 2058)
StartAngle Property

Description
Defines or returns the start of the object for the CircularArc, EllipseArc, EllipseSegment and PieSegment objects. The information is in counterclockwise direction in degrees, beginning at the 12:00 clock position.

Example:
The "PieSegmentConfiguration()" procedure accesses the properties of the Pie Segment. In this example the pie segment begins at 40° and ends at 180°:

```
Sub PieSegmentConfiguration()
    'VBA727
    Dim PieSegment As HMIPieSegment
    Set PieSegment = ActiveDocument.HMIObjects.AddHMIObject("PieSegment1", "HMIPieSegment")
    With PieSegment
        .StartAngle = 40
        .EndAngle = 180
    End With
End Sub
```

See also
EndAngle Property (Page 2222)
PieSegment Object (Page 2032)
EllipseSegment Object (Page 1963)
EllipseArc Object (Page 1960)
CircularArc Object (Page 1931)

StatusText Property

Description
Defines or returns the text that will be displayed in the status bar when you point with the mouse to a user-defined menu entry or user-defined icon.
Example:

In the following example, a user-defined menu with two menu entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line. A status bar entry will be defined for each menu entry:

```vba
Sub CreateDocumentMenus()
'VBA728
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Dim objSubMenu As HMIMenuItem

  'Add menu:
  Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")

  'Add menuitems to custom-menu:
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "My first menuitem")
  Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second menuitem")

  'Add seperator to custom-menu:
  Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")

  'Add submenu to custom-menu:
  Set objSubMenu = objDocMenu$menuItems.InsertSubMenu(4, "dSubMenu1_4", "My first submenu")

  'Add menuitems to submenu:
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "My first submenuitem")
  Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "My second submenuitem")

  'Assign statustexts to every menuitem
  With objDocMenu
    .MenuItems(1).StatusText = "My first menuitem"
    .MenuItems(2).StatusText = "My second menuitem"
    .MenuItems(4).SubMenu.Item(1).StatusText = "My first submenuitem"
    .MenuItems(4).SubMenu.Item(2).StatusText = "My second submenuitem"
  End With
End Sub
```

See also

- ToolbarItem Object (Page 2083)
- MenuItem Object (Page 2013)

SubMenu Property

Description

Returns a MenuItems listing if the specified object is the "SubMenu" type.

Use the SubMenu listing if you wish to create a submenu in a user-defined menu.
Example:

In the following example, a user-defined menu with two menu entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line:

Sub CreateDocumentMenus()
'VBA730
Dim objDocMenu As HMIMenu
Dim objMenuItem As HMIMenuItem
Dim objSubMenu As HMIMenuItem
'
'Add menu:
Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
'
'Add menuitems to custom-menu:
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "My first menuitem")
Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second menuitem")
'
'Add seperator to custom-menu:
Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")
'
'Add submenu to custom-menu:
Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "My first submenu")
'
'Add menuitems to submenu:
Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "My first submenuitem")
Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "My second submenuitem")
End Sub

See also

MenuItem Object (Page 2013)

SymbolLibraries Property

Description

Returns a SymbolLibraries listing containing objects of the "SymbolLibrary" type.
Use SymbolLibraries(1) to return the "Global Library". Use SymbolLibraries(2) to return the "Project Library".
Example:

In the following example the names of the libraries will be output:

Sub ShowSymbolLibraries()
    'VBA731
    Dim colSymbolLibraries As HMISymbolLibraries
    Dim objSymbolLibrary As HMISymbolLibrary
    Set colSymbolLibraries = Application.SymbolLibraries
    For Each objSymbolLibrary In colSymbolLibraries
        MsgBox objSymbolLibrary.Name
    Next objSymbolLibrary
End Sub

See also

Application Object (Page 1909)

T

TabOrderAlpha Property

Description

Defines or returns the position of the object in the TAB sequence for the alpha / tab order cursor.
**Example:**

In this example two I/O fields will be inserted in the active picture and the TAB sequence will then be defined:

```vba
Sub IOFieldConfig()
  'VBA734
  Dim objIOField1 As HMIIOField
  Dim objIOField2 As HMIIOField
  Set objIOField1 = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  Set objIOField2 = ActiveDocument.HMIObjects.AddHMIObject("IOField2", "HMIIOField")
  With objIOField1
    .Top = 10
    .Left = 10
    .TabOrderAlpha = 1
  End With
  With objIOField2
    .Top = 100
    .Left = 10
    .TabOrderAlpha = 2
  End With
End Sub
```

**See also**

- Document Object (Page 1949)

**TabOrderAllHMIObjects Property**

**Description**

TRUE if all the objects in a picture are to be included in the configured TAB sequence. BOOLEAN write-read access.
Example:

The "ConfigureTabOrder()" procedure defines which objects in the active picture are to be included in the configured TAB sequence. In this example all the objects will be included in the TAB sequence:

Sub ConfigureTabOrder()
'VBA733
With ActiveDocument
 .TABOrderAllHMIObjects = True
 .TABOrderKeyboard = False
 .TABOrderMouse = False
 .TABOrderOtherAction = False
End With
End Sub

See also

TabOrderByOtherAction Property (Page 2432)
TabOrderByMouse Property (Page 2432)
TabOrderByKeyboard Property (Page 2431)
Document Object (Page 1949)

TabOrderByKeyboard Property

Description

TRUE if objects with a keyboard operation event configured to them are to be included in the configured TAB sequence. BOOLEAN write-read access.

Example:

The "ConfigureTabOrder()" procedure defines which objects in the active picture are to be included in the configured TAB sequence. In this example objects with a keyboard operation will be included in the TAB sequence:

Sub ConfigureTabOrder()
'VBA735
With ActiveDocument
 .TABOrderAllHMIObjects = True
 .TABOrderKeyboard = False
 .TABOrderMouse = False
 .TABOrderOtherAction = False
End With
End Sub
See also

TabOrderOtherAction Property (Page 2432)
TabOrderMouse Property (Page 2432)
TabOrderAllHMIObjects Property (Page 2430)
Document Object (Page 1949)

TabOrderMouse Property

Description

TRUE if objects with a mouse operation event configured to them are to be included in the configured TAB sequence. BOOLEAN write-read access.

Example:

The "ConfigureTabOrder()" procedure defines which objects in the active picture are to be included in the configured TAB sequence. In this example objects with a mouse operation event will be included in the TAB sequence:

Sub ConfigureTabOrder()
  'VBA736
  With ActiveDocument
    .TABOrderAllHMIObjects = True
    .TABOrderKeyboard = False
    .TABOrderMouse = False
    .TABOrderOtherAction = False
  End With
End Sub

See also

TabOrderOtherAction Property (Page 2432)
TabOrderKeyboard Property (Page 2431)
TabOrderAllHMIObjects Property (Page 2430)
Document Object (Page 1949)

TabOrderOtherAction Property

Description

TRUE if objects with an event other than a mouse or keyboard operation event configured to them are to be included in the configured TAB sequence. BOOLEAN write-read access.
Example:

The "ConfigureTabOrder()" procedure defines which objects in the active picture are to be included in the configured TAB sequence. In this example objects with events other than a mouse or keyboard operation will be included in the TAB sequence:

Sub ConfigureTabOrder()
'VBA737
With ActiveDocument
  .TABOrderAllHMIObjects = True
  .TABOrderKeyboard = False
  .TABOrderMouse = False
  .TABOrderOtherAction = False
End With
End Sub

See also

TabOrderMouse Property (Page 2432)
TabOrderKeyboard Property (Page 2431)
TabOrderAllHMIObjects Property (Page 2430)
Document Object (Page 1949)

TabOrderSwitch Property

Description

Defines or returns the position of the object in the TAB sequence.
Example:

In this example two I/O fields will be inserted in the active picture and the TAB sequence will then be defined:

```
Sub IOFieldConfig()
  'VBA732
  Dim objIOField1 As HMIIOField
  Dim objIOField2 As HMIIOField
  Set objIOField1 = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
  Set objIOField2 = ActiveDocument.HMIObjects.AddHMIObject("IOField2", "HMIIOField")
  With objIOField1
    .Top = 10
    .Left = 10
    .TabOrderSwitch = 1
  End With
  With objIOField2
    .Top = 100
    .Left = 10
    .TabOrderSwitch = 2
  End With
End Sub
```

See also

HMIOBJECT Object (Page 1988)

Tag Property

Description

Defines or returns information text for a user-defined menu entry or user-defined icon. You can use the Tag property for example to briefly describe what the menu entry does.
Example:

In the following example, a user-defined menu with two menus entries and a submenu with two entries will be created in the active picture. The submenu will be visually distinguished by a dividing line:

```vba
Sub CreateDocumentMenus()
    'VBA738
    Dim objDocMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem
    Dim objSubMenu As HMIMenuItem
    'Add menu:
    Set objDocMenu = ActiveDocument.CustomMenus.InsertMenu(1, "DocMenu1", "Doc_Menu_1")
    'Add menuitems to custom-menu:
    Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(1, "dmItem1_1", "My first menuitem")
    Set objMenuItem = objDocMenu.MenuItems.InsertMenuItem(2, "dmItem1_2", "My second menuitem")
    'Add seperator to custom-menu:
    Set objMenuItem = objDocMenu.MenuItems.InsertSeparator(3, "dSeparator1_3")
    'Add submenu to custom-menu:
    Set objSubMenu = objDocMenu.MenuItems.InsertSubMenu(4, "dSubMenu1_4", "My first submenu")
    'Add menuitems to submenu:
    Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(5, "dmItem1_5", "My first submenuitem")
    Set objMenuItem = objSubMenu.SubMenu.InsertMenuItem(6, "dmItem1_6", "My second submenuitem")
    'To place an additional information:
    With objDocMenu
        .MenuItems(1).Tag = "This is the first menuitem"
    End With
End Sub
```

See also
- ToolbarItem Object (Page 2083)
- MenuItem Object (Page 2013)

Tag property

Description
Is used for the "Graphic Object Update Wizard" tool and is not evaluated for the "HMIAdvancedAnalogDisplay" and "HMIAdvancedStateAnalogDisplay" objects.
tagname property

Description
Is used for the "Graphic Object Update Wizard" tool and is not evaluated for the "HMIAdvancedAnalogDisplay" and "HMIAdvancedStateAnalogDisplay" objects.

TagPrefix Property

Description
Defines or returns the tag prefix for all the tags contained in the Picture Window object.

Example:
The picture "InputOutput" is to be displayed in the picture window. The picture "InputOutput" contains three I/O fields which are linked to a structure tag. The structure tag consists of the elements EA1, EA2, EA3; one element each for each I/O field.

Three such structure tags have been define in the project, with structure names Struct1, Struct2 and Struct3.

The tag prefix is in this case the structure name followed by a period. Specify the tag prefix as, say, Struct2. (the period is necessary in order to address the elements of the structure tag as structure elements in a syntactically correct way). The I/O fields in the picture "InputOutput" are then linked to the elements in structure tag Struct2:

Tag Prefix: "Struct2."

- Output value (first I/O field): EA1
- Output value (second I/O field): EA2
- Output value (third I/O field): EA3

The current tag connection in the picture window is then

- Output value (first I/O field): Struct2.EA1
- Output value (second I/O field): Struct2.EA2
- Output value (third I/O field): Struct2.EA3
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be configured:

Sub PictureWindowConfig()
'VBA739
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
  .AdaptPicture = False
  .AdaptSize = False
  .Caption = True
  .CaptionText = "Picturewindow in runtime"
  .OffsetLeft = 5
  .OffsetTop = 10
  'Replace the picturename "Test.PDL" with the name of an existing document from your "GraCS"-Folder of your active project
  .PictureName = "Test.PDL"
  .ScrollBars = True
  .ServerPrefix = "my_Server::"
  .TagPrefix = "Struct."
  .UpdateCycle = 5
  .Zoom = 100
End With
End Sub

See also

PictureWindow Object (Page 2029)

TagScaleParam1 property

Description

Sets the value1 for the value range process.

TagScaleParam2 property

Description

Sets the value2 for the value range process.
TagScaleParam3 property

Description
Sets the value3 for the value range process.

TagScaleParam4 property

Description
Sets the value4 for the value range process.

TagStartvaluePersistence property

Description
Defines whether an internal tag is set as persistent. You can only set internal tags as persistent.

tagtype property

Description
Is used for the "Graphic Object Update Wizard" tool and is not evaluated for the "HMIAdvancedAnalogDisplay" and "HMIAdvancedStateAnalogDisplay" objects.

Template property

Description
Returns the template for displaying the window content of the "ApplicationWindow" object. Read only access.
The "ApplicationWindow" object can be supplied from applications of the Global Script and the report system:

<table>
<thead>
<tr>
<th>GSC Diagnostics</th>
<th>The application window is supplied by applications of the Global Script. The results of the diagnostics system are displayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSC Runtime</td>
<td>The application window is supplied by applications of the Global Script. The analysis results regarding characteristics in Runtime are displayed.</td>
</tr>
<tr>
<td>All Jobs</td>
<td>The application window is supplied by the report system. The available reports are displayed as a list.</td>
</tr>
<tr>
<td>All Jobs – Shortcut Menu</td>
<td>The application window is supplied by the report system. The available reports are displayed as a list. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the report.</td>
</tr>
<tr>
<td>Job Detail View</td>
<td>The application window is supplied by the report system. The available reports are displayed in a selection menu. Detailed information is displayed for the selected report.</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Selected Jobs - Shortcut Menu</td>
<td>The application window is supplied by the report system. The available reports are displayed as a list. This list only contains reports which you have activated the option &quot;Mark for print job list&quot; in the &quot;Print Job Properties&quot; dialog. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the report.</td>
</tr>
</tbody>
</table>

See also

ApplicationWindow Object (Page 1911)

Text Property

Description

Defines or returns the labeling for an object.

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the label will be defined:

```vba
Sub ButtonConfiguration()
'VBA740
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjcts.AddHMIObject("Button1", "HMIButton")
With objButton
  .Text = "Button1"
End With
End Sub
```

See also

Button Object (Page 1921)
StaticText Object (Page 2067)
OptionGroup Object (Page 2025)
CheckBox Object (Page 1924)

TextBiblIDs property

Description

Only used internally.
See also

TitleBackColorActiveEnd property

Description
Only used internally.

See also

TitleBackColorActiveStart property

Description
Only used internally.

See also

TitleBackColorInactiveEnd property

Description
Only used internally.

See also

TitleBackColorInactiveStart property

Description
Only used internally.

See also

TextList Object (Page 2076)

See also

PictureWindow Object (Page 2029)
TitleForeColorActive property

Description
Only used internally.

See also
PictureWindow Object (Page 2029)

TitleForeColorInactive property

Description
Only used internally.

See also
PictureWindow Object (Page 2029)

Toggle Property

Description
TRUE, if the button or round button should lock after being operated in Runtime. BOOLEAN write-read access.

Example:
The "RoundButtonConfiguration()" procedure accesses the properties of the RoundButton. In this example the round button is intended to latch down when pressed in Runtime:

```vba
Sub RoundButtonConfiguration()
  'VBA741
  Dim objRoundButton As HMIRoundButton
  Set objRoundButton = ActiveDocument.HMIOObjects.AddHMIObject("RButton1", "HMIRoundButton")
  With objRoundButton
    .Toggle = True
  End With
End Sub
```

See also
RoundButton Object (Page 2052)
ToleranceHigh Property

Description

Defines or returns the limit value for “Tolerance high”.
The type of the evaluation (in percent or absolute) is defined in the TypeToleranceHigh property.
Monitoring of the limit value only takes effect when the CheckToleranceHigh property is set to “True”.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the limit values. In this example the limit value for “Tolerance High” will be configured:

```vba
Sub BarGraphLimitConfiguration()
  'VBA742
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis = absolute
    .TypeToleranceHigh = False
    'Activate monitoring
    .CheckToleranceHigh = True
    'Set barcolor = "yellow"
    .ColorToleranceHigh = RGB(255, 255, 0)
    'Set upper limit to "40"
    .ToleranceHigh = 40
  End With
End Sub
```

See also

TypeToleranceHigh Property (Page 2456)
CheckToleranceHigh Property (Page 2176)
BarGraph Object (Page 1914)

ToleranceLow Property

Description

Defines or returns the limit value for “Tolerance low”.
The type of the evaluation (in percent or absolute) is defined in the TypeToleranceLow property.
Monitoring of the limit value only takes effect when the CheckToleranceLow property is set to “True”.

See also

TypeToleranceLow Property (Page 2456)
CheckToleranceLow Property (Page 2176)
BarGraph Object (Page 1914)
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the limit values. In this example the limit value for "Tolerance Low" will be configured.

Sub BarGraphLimitConfiguration()
    ' VBA743
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        ' Set analysis = absolute
        .TypeToleranceLow = False
        ' Activate monitoring
        .CheckToleranceLow = True
        ' Set barcolor = "red"
        .ColorToleranceLow = RGB(255, 0, 0)
        ' Set lower limit to "40"
        .ToleranceLow = 40
    End With
End Sub

See also

TypeToleranceLow Property (Page 2457)
CheckToleranceLow Property (Page 2177)
BarGraph Object (Page 1914)

ToolbarItems Property

Description

Returns a listing containing all the elements (icons and separation lines) of a user-defined toolbar.
Example

In the following example a user-defined toolbar with two icons is created in the active picture. These icons are separated by a dividing line:

Sub AddDocumentSpecificCustomToolbar()
    'VBA744
    Dim objToolbar As HMIToolbar
    Dim objToolbarItem As HMIToolbarItem
    Set objToolbar = ActiveDocument.CustomToolbars.Add("DocToolbar")
    
    'Add symbol-icon to userdefined toolbar
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(1, "tItem1_1", "My first symbol-icon")
    Set objToolbarItem = objToolbar.ToolbarItems.InsertToolbarItem(3, "tItem1_3", "My second symbol-icon")
    Set objToolbarItem = objToolbar.ToolbarItems.InsertSeparator(2, "tSeparator1_2")
End Sub

See also

ToolbarItem Object (Page 2083)
Toolbar Object (Page 2080)

ToolbarItemType property

Description

Returns the type of the "HMIToolbarItem" object of a user-defined toolbar as a "string".

<table>
<thead>
<tr>
<th>Returned Value</th>
<th>Type in the toolbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Separator</td>
</tr>
<tr>
<td>1</td>
<td>Icon</td>
</tr>
</tbody>
</table>

Example

In the following example the type of the first object in the first user-defined toolbar in the active picture is output:

Sub ShowFirstObjectOfCollection()
    'VBA353
    Dim strType As String
    strType = ActiveDocument.CustomToolbars(1).ToolbarItems(1).ToolbarItemType
    MsgBox strType
End Sub
ToolTipText Property

Description

Defines or returns the text that will be displayed as a Tooltip when you run the mouse over an object (HMIObject, icon).

Example:

The "RectangleConfiguration()" procedure accesses the properties of the Rectangle object. In this example a tool tip text will be assigned to the rectangle:

```vba
Sub RectangleConfiguration()
  'VBA745
  Dim objRectangle As HMIRectangle
  Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
  With objRectangle
    .ToolTipText = "This is a rectangle"
  End With
End Sub
```

The following example shows how you have to initialize the property prior to dynamization:

```vba
Sub Dyn()
  'VBA823
  Dim objCircle As HMICircle
  Dim doc As Document
  Dim objDynDialog As HMIDynamicDialog
  Set doc = ActiveDocument
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle","HMICircle")
  objCircle.ObjectName = "Circle1"
  objCircle.BorderColor = RGB(255, 0, 0)
  objCircle.BackColor = RGB(0, 255, 0)
  objCircle.ToolTipText = "Text"
  Set objDynDialog = objCircle.ToolTipText.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'Var'")
End Sub
```

See also

ToolbarItem Object (Page 2083)

HMIObject Object (Page 1988)

How to dynamize a property with the Dynamic dialog (Page 1691)
Top Property

Description

Defines or returns the Y-coordinate of an object (measured from the top left edge of the picture) in pixels. The Y-coordinate relates to the top left corner of the rectangle enclosing the object.

Example:

The "RectangleConfiguration()" procedure accesses the properties of the Rectangle object. In this example the rectangle will be set to position 10/40:

```vba
Sub RectangleConfiguration()
    'VBA746
    Dim objRectangle As HMIRectangle
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
    With objRectangle
        .Left = 10
        .Top = 40
    End With
End Sub
```

See also

View Object (Page 2102)
HMIObject Object (Page 1988)

TopConnectedObjectName Property

Description

Returns the name of the end object to which the connector is connected. Read only access.

Example:

An example showing how to use the TopConnectedObjectName property can be found in this documentation under the heading "ObjConnection Object".

See also

ObjConnection object (Page 2019)
TopConnectedConnectionPointIndex Property

Description

Returns the connection point on the object to which the connector is connected.

<table>
<thead>
<tr>
<th>Connection Point</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
<td>0</td>
</tr>
<tr>
<td>Right</td>
<td>1</td>
</tr>
<tr>
<td>Down</td>
<td>2</td>
</tr>
<tr>
<td>Left</td>
<td>3</td>
</tr>
</tbody>
</table>

Example:

An example showing how to use the BottomConnectedObjectName property can be found in this documentation under the heading "ObjConnection Object".

See also

ObjConnection object (Page 2019)

Transparency property

Description

Defines the degree of transparency of the object display. Values between 0 and 100 indicate the transparency as a percentage. In the case of a semi-transparent object, other objects shine through. A 100% transparent object is invisible. An invisible object can also be controlled in Runtime.
Example

Sub addTransparentObject()
    Dim objHMICircle As HMICircle
    Set objHMICircle = ActiveDocument.HMIObjects.AddHMIObject("Circle", "HMICircle")
    objHMICircle.Transparency = 40
End Sub

Trend Property

Description

TRUE if the trend or tendency of the measured value being monitored (rising or falling) is to be indicated by a little arrow. BOOLEAN write-read access.

Example:

The "BarGraphConfiguration()" procedure configures In this example the trend of the measured value will be indicated:

Sub BarGraphConfiguration()
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
    With objBarGraph
        .trend = True
    End With
End Sub

See also

BarGraph Object (Page 1914)

trend property

Description

Is used for the "Graphic Object Update Wizard" tool and is not evaluated for the "HMIAdvancedAnalogDisplay" and "HMIAdvancedStateAnalogDisplay" objects.
TrendColor Property

Description

Defines or returns the color of the trend display.

The trend display indicates the tendency (rising or falling) of the measuring value being monitored by a small arrow. In order to activate the trend display, the Trend property must be set to "True". LONG write-read access.

Determination of Color Value

The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:

The "BarGraphConfiguration()" procedure configures In this example the trend in the measured value will be indicated. The trend display will be set to "Red":

```vba
Sub BarGraphConfiguration()
'VBA748
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
 .trend = True
 .TrendColor = RGB(255, 0, 0)
End With
End Sub
```

See also

- Trend Property (Page 2448)
- BarGraph Object (Page 1914)

Trigger Property

Description

Returns a Trigger object. Use the Trigger property when making a property dynamic with the aid of a script.
Example:

In this example the "Radius" property of a circle will be made dynamic with the aid of a C script (the output value sets the radius):

```vba
Sub AddDynamicAsCSkriptToProperty()
    'VBA749
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMI_Circle

    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMI_Circle")
    Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
    With objVBScript
        .Trigger.Type = hmiTriggerTypeStandardCycle
        .Trigger.CycleType = hmiCycleType_2s
        .Trigger.Name = "Trigger1"
    End With
End Sub
```

See also

- Trigger Object (Page 2087)
- ScriptInfo Object (Page 2058)

Type Property

Description

Returns or defines the type of an object.

The object type is returned as either a string or and integer.

Example:

The "RectangleConfiguration()" procedure accesses the properties of the Rectangle object. In this example the object type will be output:

```vba
Sub RectangleConfiguration()
    'VBA750
    Dim objRectangle As HMI_Rectangle
    Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMI_Rectangle")
    With objRectangle
        MsgBox "Objecttype: " & .Type
    End With
End Sub
```
See also

- Trigger Object (Page 2087)
- ToolbarItem Object (Page 2083)
- SourceLink Object (Page 2065)
- Property Object (Page 2042)
- HMIObject Object (Page 1988)
- FolderItem Object (Page 1972)
- DestLink Object (Page 1945)

TypeAlarmHigh Property

Description

TRUE, when the upper limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "50".

```vba
Sub BarGraphLimitConfiguration()
'VBA751
Dim objBarGraph As HMIbarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIbarGraph")
With objBarGraph
  'Set analysis = absolute
  .TypeAlarmHigh = False
  'Activate monitoring
  .CheckAlarmHigh = True
  'Set barcolor = "yellow"
  .ColorAlarmHigh = RGB(255, 255, 0)
  'Set upper limit = "50"
  .AlarmHigh = 50
End With
End Sub
```

See also

- ColorAlarmHigh Property (Page 2182)
- CheckAlarmHigh Property (Page 2170)
TypeAlarmLow Property

Description
TRUE, when the lower limit value, at which an alarm is triggered, should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

Example:
The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "10".

Sub BarGraphLimitConfiguration()
'VBA752
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis = absolute
  .TypeAlarmLow = False
  'Activate monitoring
  .CheckAlarmLow = True
  'Set barcolor = "yellow"
  .ColorAlarmLow = RGB(255, 255, 0)
  'Set lower limit = "10"
  .AlarmLow = 10
End With
End Sub

See also
ColorAlarmLow Property (Page 2183)
CheckAlarmLow Property (Page 2171)
AlarmLow Property (Page 2115)
BarGraph Object (Page 1914)
TypeLimitHigh4 Property

Description

TRUE, when the "Reserve 4" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "70".

Sub BarGraphLimitConfiguration()
'VBA753
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjec.ts.AddHMIObjec.t("Bar1", "HMIBarGraph")
With objBarGraph
  'Set analysis = absolute
  .TypeLimitHigh4 = False
  'Activate monitoring
  .CheckLimitHigh4 = True
  'Set barcolor = "red"
  .ColorLimitHigh4 = RGB(255, 0, 0)
  'Set upper limit = "70"
  .LimitHigh4 = 70
End With
End Sub

See also

LimitHigh4 Property (Page 2307)
CheckLimitHigh4 Property (Page 2172)
BarGraph Object (Page 1914)

TypeLimitHigh5 Property

Description

TRUE, when the "Reserve 5" upper limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "80".

Sub BarGraphLimitConfiguration()
'VBA754
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObj ect. AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis = absolute
.TypeLimitHigh5 = False
'Activate monitoring
.CheckLimitHigh5 = True
'Set barcolor = "black"
.ColorLimitHigh5 = RGB(0, 0, 0)
'Set upper limit = "70"
.LimitHigh5 = 70
End With
End Sub

See also

LimitHigh5 Property (Page 2307)
CheckLimitHigh5 Property (Page 2173)
BarGraph Object (Page 1914)

TypeLimitLow4 Property

Description

TRUE, when the "Reserve 4" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
**Example:**

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "5".

```vba
Sub BarGraphLimitConfiguration()
  'VBA755
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis = absolute
    .TypeLimitLow4 = False
    'Activate monitoring
    .CheckLimitLow4 = True
    'Set barcolor = "green"
    .ColorLimitLow4 = RGB(0, 255, 0)
    'Set lower limit = "5"
    .LimitLow4 = 5
  End With
End Sub
```

**See also**

- LimitLow4 Property (Page 2308)
- ColorLimitLow4 Property (Page 2187)
- CheckLimitLow4 Property (Page 2174)
- BarGraph Object (Page 1914)

**TypeLimitLow5 Property**

**Description**

TRUE, when the "Reserve 5" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "0".

```vba
Sub BarGraphLimitConfiguration()
'VBA756
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIOBJECT("Bar1", "HMIBarGraph")
With objBarGraph
' Set analysis = absolute
.TypeLimitLow5 = False
' Activate monitoring
.CheckLimitLow5 = True
' Set barcolor = "white"
.ColorLimitLow5 = RGB(255, 255, 255)
' Set lower limit = "0"
.LimitLow5 = 0
End With
End Sub
```

See also

- LimitLow5 Property (Page 2309)
- ColorLimitLow5 Property (Page 2188)
- CheckLimitLow5 Property (Page 2175)
- BarGraph Object (Page 1914)

TypeToleranceHigh Property

Description

TRUE, when the "Tolerance high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the limit values. In this example the limit value for "Tolerance High" will be configured:

```vba
Sub BarGraphLimitConfiguration()
  'VBA757
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    .TypeToleranceHigh = False
    .CheckToleranceHigh = True
    .ColorToleranceHigh = RGB(255, 255, 0)
    .ToleranceHigh = 40
  End With
End Sub
```

See also

- ColorToleranceHigh Property (Page 2189)
- CheckToleranceHigh Property (Page 2176)
- BarGraph Object (Page 1914)

TypeToleranceLow Property

Description

TRUE, when the "Tolerance low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the limit values. In this example the limit value for "Tolerance Low" will be configured:

```vba
Sub BarGraphLimitConfiguration()
'VBA758
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
  .TypeToleranceLow = False
  .CheckToleranceLow = True
  .ColorToleranceLow = RGB(255, 0, 0)
  .ToleranceLow = 10
End With
End Sub
```

See also

- ToleranceLow Property (Page 2442)
- ColorToleranceLow Property (Page 2190)
- CheckToleranceLow Property (Page 2177)
- BarGraph Object (Page 1914)

TypeWarningHigh Property

Description

TRUE, when the "Warning high" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "75".

```vba
Sub BarGraphLimitConfiguration()
'VBA759
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis = absolute
.TypeWarningHigh = False
'Activate monitoring
.CheckWarningHigh = True
'Set barcolor = "red"
.ColorWarningHigh = RGB(255, 0, 0)
'Set upper limit = "75"
.WarningHigh = 75
End With
End Sub
```

See also

- WarningHigh Property (Page 2552)
- ColorWarningHigh Property (Page 2192)
- CheckWarningHigh Property (Page 2178)
- BarGraph Object (Page 1914)

TypeWarningLow Property

Description

TRUE, when the "Warning low" lower limit value should be evaluated as a percentage. FALSE, when the evaluation should be as an absolute value. BOOLEAN write-read access.
Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "12".

Sub BarGraphLimitConfiguration()
'VBA760
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis = absolute
.TypeWarningLow = False
'Activate monitoring
.CheckWarningLow = True
'Set barcolor = "magenta"
.ColorWarningLow = RGB(255, 0, 255)
'Set lower limit = "12"
.WarningLow = 12
End With
End Sub

See also

WarningLow Property (Page 2553)
ColorWarningLow Property (Page 2193)
CheckWarningLow Property (Page 2179)
BarGraph Object (Page 1914)

U

Underlined Property

Description

TRUE if the font attribute "Underline" is set for the language-dependent text in the object. BOOLEAN write-read access.
Example:

The following example sets the font attributes of a button for French and English:

Sub ExampleForLanguageFonts()
  Dim colLangFonts As HMILanguageFonts
  Dim objButton As HMIButton
  Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
  objButton.Text = "DefText"
  Set colLangFonts = objButton.LDFonts
  'Set font-properties for french:
  With colLangFonts.ItemByLCID(1036)
    .Family = "Courier New"
    .Bold = True
    .Italic = False
    .Underlined = True
    .Size = 12
  End With
  'Set font-properties for english:
  With colLangFonts.ItemByLCID(1033)
    .Family = "Times New Roman"
    .Bold = False
    .Italic = True
    .Underlined = False
    .Size = 14
  End With
End Sub

See also

Size Property (Page 2419)
Parent Property (Page 2359)
LanguageID Property (Page 2289)
Italic Property (Page 2282)
Family Property (Page 2229)
Bold Property (Page 2151)
Application Property (Page 2121)
LanguageFont Object (Page 1996)
UnselBGColor Property

Description
Defines or returns the background color of entries in the text list object which are not selected. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)

Example:
The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the colors will be defined for entries that are not selected in the selection list:

Sub TextListConfiguration()
'VBA762
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
.UnselBGColor = RGB(255, 0, 0)
.UnselTextColor = RGB(0, 0, 0)
End With
End Sub

See also
TextList Object (Page 2076)

UnselTextColor Property

Description
In the case of the TextList object, defines or returns the color of text in the selection list for entries that are not selected. LONG write-read access.

Determination of Color Value
The color is displayed in RGB format (Red, Green, Blue). Specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255).

Use VBA function "RGB" to assign a color to a property. The color "red", for example, is represented as follows: RGB(255, 0, 0)
Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the colors will be defined for entries that are not selected in the selection list:

```vba
Sub TextListConfiguration()
  'VBA763
  Dim objTextList As HMITextList
  Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
  With objTextList
    .UnselBGColor = RGB(255, 0, 0)
    .UnselTextColor = RGB(0, 0, 0)
  End With
End Sub
```

See also

TextList Object (Page 2076)

UpdateCycle Property

Description

Defines or returns the type and frequency of updates to the picture window in Runtime.

<table>
<thead>
<tr>
<th>Update Cycle</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upon change</td>
<td>0</td>
</tr>
<tr>
<td>250 ms</td>
<td>1</td>
</tr>
<tr>
<td>500 ms</td>
<td>2</td>
</tr>
<tr>
<td>1 s</td>
<td>3</td>
</tr>
<tr>
<td>2 s</td>
<td>4</td>
</tr>
<tr>
<td>5 s</td>
<td>5</td>
</tr>
<tr>
<td>10 s</td>
<td>6</td>
</tr>
<tr>
<td>1 min</td>
<td>7</td>
</tr>
<tr>
<td>5 min</td>
<td>8</td>
</tr>
<tr>
<td>10 min</td>
<td>9</td>
</tr>
<tr>
<td>1 h</td>
<td>10</td>
</tr>
<tr>
<td>User cycle 1</td>
<td>11</td>
</tr>
<tr>
<td>User cycle 2</td>
<td>12</td>
</tr>
<tr>
<td>User cycle 3</td>
<td>13</td>
</tr>
<tr>
<td>User cycle 4</td>
<td>14</td>
</tr>
<tr>
<td>User cycle 5</td>
<td>15</td>
</tr>
<tr>
<td>Picture cycle</td>
<td>255</td>
</tr>
</tbody>
</table>
Example:

The "PictureWindowConfig" procedure accesses the properties of the picture window. In this example the picture window will be updated every 5 seconds in Runtime:

```vba
Sub PictureWindowConfig()
'VBA764
Dim objPicWindow As HMIPictureWindow
Set objPicWindow = ActiveDocument.HMIObjects.AddHMIObject("PicWindow1", "HMIPictureWindow")
With objPicWindow
 .UpdateCycle = 5
End With
End Sub
```

See also

PictureWindow Object (Page 2029)

**UseEventState property**

**Description**

Specifies for the "HMIAdvancedStateDisplay" object whether the group value is evaluated for the representation of the states.

If the group value is used, you can assign pictures for the individual alarm statuses.

**UsedLanguage property**

**Description**

Use the UsedLanguage property to set the code page that matches the character set used.

LONG write-read access.
Example

The "UsedLanguage" property and language ID "1033" are used in the following example to set the code page to English US.

```
Sub AddDynamicAsCSkriptToProperty()
  'VBA856
  Dim objCScript As HMIScriptInfo
  Dim objCircle As HMICircle
  Dim strCode As String
  strCode = "long lHeight;" & vbCrLf & "int check;" & vbCrLf
  strCode = strCode & "GetHeight (""events.PDL"", ""myCircle"""); & vbcrlf
  strCode = strCode & "lHeight = lHeight+5;" & vbCrLf
  strCode = strCode & "check = SetHeight(""events.PDL"", ""myCircle"", lHeight");"
  strCode = strCode & vbCrLf & "//Return-Type: BOOL" & vbCrLf
  strCode = strCode & "return check;"
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_C", "HMICircle")
  'Create dynamic for Property "Radius":
  Set objCScript = objCircle.Height.CreateDynamic(hmiDynamicCreationTypeCScript)
  'set Sourcecode and cycletime:
  With objCScript
    .SourceCode = strCode
    .Trigger.Type = hmiTriggerTypeStandardCycle
    .Trigger.CycleType = hmiCycleType_2s
    .Trigger.Name = "Trigger1"
  'Set language English-US
  .UsedLanguage = 1033
  End With
End Sub
```

**UseGlobalAlarmClasses property**

**Description**

Defines whether to use globally configured alarm classes to visualize message events. The property is only relevant for PCS7 projects.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Activates the global settings made in PCS7 alarm editor for visualizing the message events.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Visualization of the message events is defined locally for each message class.</td>
</tr>
</tbody>
</table>
**UseGlobalSettings property**

**Description**

Specify whether to use global settings to assign message events to the buttons visualized in the group view. The display of the message events is configured using the "MessageClass" properties. The property is only relevant for PCS7 projects.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Activates the settings made in the PCS7 alarm editor for the assignment of message events to the buttons in the group display. The bit numbers in the group value are assigned to the respective buttons.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The message types are assigned locally to the buttons in the group display.</td>
</tr>
</tbody>
</table>

**UserValue1 Property**

**Description**

Defines or returns any value in the case of the GroupDisplay object. The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

**Example:**

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example four different user values will be assigned:

```vba
Sub GroupDisplayConfiguration()
'VBA765
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .UserValue1 = 0
  .UserValue2 = 25
  .UserValue3 = 50
  .UserValue4 = 75
End With
End Sub
```

**See also**

UserValue4 Property (Page 2468)
UserValue3 Property (Page 2467)
UserValue2-Eigenschaft

Description
Defines or returns any value in the case of the GroupDisplay object.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.

Example:
The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example four different user values will be assigned:

```vba
Sub GroupDisplayConfiguration()
    'VBA766
    Dim objGroupDisplay As HMIGroupDisplay
    Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
    With objGroupDisplay
        .UserValue1 = 0
        .UserValue2 = 25
        .UserValue3 = 50
        .UserValue4 = 75
    End With
End Sub
```

See also
UserValue4 Property (Page 2468)
UserValue3 Property (Page 2467)
UserValue1 Property (Page 2466)
GroupDisplay Object (Page 1980)

UserValue3 Property

Description
Defines or returns any value in the case of the GroupDisplay object.
The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example four different user values will be assigned:

```vba
Sub GroupDisplayConfiguration()
'VBA767
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
  .UserValue1 = 0
  .UserValue2 = 25
  .UserValue3 = 50
  .UserValue4 = 75
End With
End Sub
```

See also

- UserValue4 Property (Page 2468)
- UserValue2-Eigenschaft (Page 2467)
- UserValue1 Property (Page 2466)
- GroupDisplay Object (Page 1980)

UserValue4 Property

Description

Defines or returns any value in the case of the GroupDisplay object.

The value can be evaluated by a script, for example. This value is neither evaluated nor displayed in Runtime.
Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example four different user values will be assigned:

```vba
Sub GroupDisplayConfiguration()
'VBA768
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1", "HMIGroupDisplay")
With objGroupDisplay
   .UserValue1 = 0
   .UserValue2 = 25
   .UserValue3 = 50
   .UserValue4 = 75
End With
End Sub
```

See also

UserValue3 Property (Page 2467)
UserValue2-Eigenschaft (Page 2467)
UserValue1 Property (Page 2466)
GroupDisplay Object (Page 1980)

UseValueText property

Description

Specifies whether a text tag is used instead of a formatted analog value.

V

Value

Value Property

Description

Returns or defines the value of an object property.
Example:

Use the Value property if you wish to return or define a value with the aid of the Properties listing. In this example the property of an ActiveX Control will be accessed via the Value property:

```vba
Sub AddActiveXControl()
'VBA769
Dim objActiveXControl As HMIActiveXControl
Set objActiveXControl = ActiveDocument.HMIObjects.AddActiveXControl("WinCC_Gauge2", "XGAUGE.XGaugeCtrl.1")
'Move ActiveX-Control:
objActiveXControl.Top = 40
objActiveXControl.Left = 60
'Modify individual properties:
obActiveXControl.Properties("BackColor").value = RGB(255, 0, 0)
End Sub
```

See also

Property Object (Page 2042)

**VALUE_ACCESS_FAULT Property**

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Access to tag not permitted" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle is given dynamics with the The dynamization takes place be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        'Activate analysis of variable state
        .VariableStateChecked = True
    End With
    With objDynDialog.VariableStateValues(1)
        'define a value for every state:
        .VALUE_ACCESS_FAULT = 20
        .VALUE_ADDRESS_ERROR = 30
        .VALUE_CONVERSION_ERROR = 40
        .VALUE_HANDSHAKE_ERROR = 60
        .VALUE_HARDWARE_ERROR = 70
        .VALUE_INVALID_KEY = 80
        .VALUE_MAX_LIMIT = 90
        .VALUE_MAX_RANGE = 100
        .VALUE_MIN_LIMIT = 110
        .VALUE_MIN_RANGE = 120
        .VALUE_NOT_ESTABLISHED = 130
        .VALUE_SERVERDOWN = 140
        .VALUE_STARTUP_VALUE = 150
        .VALUE_TIMEOUT = 160
    End With
End Sub
```

See also

- VALUE_MAX_LIMIT Property (Page 2506)
- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
VALUE_ADDRESS_ERROR Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Addressing error" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA771
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMIcircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMIcircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        ,
        'Activate analysis of variablestate
        .VariableStateChecked = True
    End With
    With objDynDialog.VariableStateValues(1)
        ,
        'define a value for every state:
        .VALUE_ACCESS_FAULT = 20
        .VALUE_ADDRESS_ERROR = 30
        .VALUE_CONVERSION_ERROR = 40
        .VALUE_HANDSHAKE_ERROR = 60
        .VALUE_HARDWARE_ERROR = 70
        .VALUE_INVALID_KEY = 80
        .VALUE_MAX_LIMIT = 90
        .VALUE_MAX_RANGE = 100
        .VALUE_MIN_LIMIT = 110
        .VALUE_MIN_RANGE = 120
        .VALUE_NOT_ESTABLISHED = 130
        .VALUE_SERVERDOWN = 140
        .VALUE_STARTUP_VALUE = 150
        .VALUE_TIMEOUT = 160
    End With
End Sub

See also

VariableStateChecked Property (Page 2545)
VALUE_TIMEOUT Property (Page 2520)
VALUE_STARTUP_VALUE Property (Page 2518)
VALUE_SERVERDOWN Property (Page 2516)
VALUE_NOT_ESTABLISHED Property (Page 2514)
VALUE_MIN_RANGE Property (Page 2512)
VALUE_MIN_LIMIT Property (Page 2510)
VALUE_BAD_COMMLUV Property

Description

Specifies the value assigned to a dynamized property if quality code "bad, no communication (last usable value)" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA818
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  'define a value for every state:
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSEVR = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

- VALUE_BAD_DEVICE Property (Page 2480)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_BAD_COMMNUV Property

Description

Specifies the value assigned to a dynamized property if quality code "bad, no communication (last usable value)" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA770
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMI Circle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .QualityCodeStateChecked = True
  End With
  With objDynDialog.QualityCodeStateValues(1)
    .VALUE_BAD_COMMLUV = 20
    .VALUE_BAD_COMMNUV = 30
    .VALUE_BAD_CONFERROR = 40
    .VALUE_BAD_DEVICE = 60
    .VALUE_BAD_MISCSTATES = 70
    .VALUE_BAD_NONSPECIFIC = 80
    .VALUE_BAD_NOTCONNECTED = 90
    .VALUE_BAD_OUTOFSERV = 100
    .VALUE_BAD_PROCRELNOM = 110
    .VALUE_BAD_PROCRELSUB = 120
    .VALUE_HIGHLIMITED = 130
    .VALUE_LOWLIMITED = 140
    .VALUE_UNCERT_ENGVHIGHLIM = 150
    .VALUE_UNCERT_ENGVLOWLIM = 160
    .VALUE_UNCERT_INITVAL = 170
    .VALUE_UNCERT_LUV = 180
    .VALUE_UNCERT_MAINTDEM = 190
    .VALUE_UNCERT_MISCSTATES = 200
    .VALUE_UNCERT_NONSPECIFIC = 210
    .VALUE_UNCERT_PROCRELNOM = 220
    .VALUE_UNCERT_PROCRELSUB = 230
    .VALUE_UNCERT_SUBSTSET = 240
  End With
End Sub
```

See also

- QualityCodeStateChecked Properties (Page 2390)
- VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

**VALUE_BAD_CONFERROR Property**

**Description**

Specifies the value assigned to a dynamized property if quality code "bad, no communication, value not accepted" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        'define a value for every state:
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```
See also

- VALUE_UNCERT_MISCSTATES Property (Page 2534)
- QualityCodeStateChecked Properties (Page 2390)
- VALUE_UNCERT_SUBSTSET Property (Page 2542)
- VALUE_UNCERT_SIMVAL Property (Page 2540)
- VALUE_UNCERT_PROCRELNOM Property (Page 2538)
- VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
- VALUE_UNCERT_MAINTDEM Property (Page 2532)
- VALUE_UNCERT_LUV Property (Page 2530)
- VALUE_UNCERT_INITVAL Property (Page 2528)
- VALUE_UNCERT_ENGVONLIM Property (Page 2526)
- VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
- VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
- VALUE_LOWLIMITED Property (Page 2504)
- VALUE_HIGHLIMITED Property (Page 2500)
- VALUE_BAD_PROCRELSUB Property (Page 2492)
- VALUE_BAD_PROCRELNOM Property (Page 2490)
- VALUE_BAD_OUTOFSERV Property (Page 2488)
- VALUE_BAD_NOTCONNECTED Property (Page 2486)
- VALUE_BAD_NONSPECIFIC Property (Page 2484)
- VALUE_BAD_MISCSTATES Property (Page 2482)
- VALUE_BAD_DEVICE Property (Page 2480)
- VALUE_BAD_COMMNUV Property (Page 2476)
- VALUE_BAD_COMMLUV Property (Page 2474)
- QualityCodeStateValue Object (Page 2044)

**VALUE_BAD_DEVICE Property**

**Description**

Specifies a value assigned to a dynamized property if quality code "bad, device failure" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjets.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSEFR = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LWONLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

- VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

VALUE_BAD_MISCSTATES Property

Description

Specifies the value assigned to a dynamized property if quality code “bad miscellaneous states” occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA770
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .Activate analysis of qualitycodestate
    .QualityCodeStateChecked = True
  End With
  With objDynDialog.QualityCodeStateValues(1)
    'define a value for every state:
    .VALUE_BAD_COMMLUV = 20
    .VALUE_BAD_COMMNUV = 30
    .VALUE_BAD_CONFERROR = 40
    .VALUE_BAD_DEVICE = 60
    .VALUE_BAD_MISCSTATES = 70
    .VALUE_BAD_NONSPECIFIC = 80
    .VALUE_BAD_NOTCONNECTED = 90
    .VALUE_BAD_OUTOFSERV = 100
    .VALUE_BAD_PROCRELNOM = 110
    .VALUE_BAD_PROCRELSUB = 120
    .VALUE_HIGHLIMITED = 130
    .VALUE_LOWLIMITED = 140
    .VALUE_UNCERT_ENGVHIGHLIM = 150
    .VALUE_UNCERT_ENGVLOWLIM = 160
    .VALUE_UNCERT_INITVAL = 170
    .VALUE_UNCERT_LUV = 180
    .VALUE_UNCERT_MAINTDEM = 190
    .VALUE_UNCERT_MISCSTATES = 200
    .VALUE_UNCERT_NONSPECIFIC = 210
    .VALUE_UNCERT_PROCRELNOM = 220
    .VALUE_UNCERT_SIMVAL = 230
    .VALUE_UNCERT_SUBSTSET = 240
  End With
End Sub
```

See also

VALUE_UNCERT_ENGVONLIM Property (Page 2526)
QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMTED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

**VALUE_BAD_NONSPECIFIC Property**

**Description**

Specifies the value assigned to a dynamized property if quality code "bad, non-specific" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSEVR = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

VALUE_UNCERT_ENGVONLIM Property (Page 2526)
QualityCodeStateChecked Properties (Page 2390)
VALUE_BAD_NOTCONNECTED Property

Description

Specifies a value assigned to a dynamized property if quality code "bad, not connected" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObj3ects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LowlIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

- VALUE_HIGHLIMITED Property (Page 2500)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_BAD_OUTOFSERV Property

Description

Specifies a value assigned to a dynamized property if quality code "bad, out of service" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA770
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
    "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .QualityCodeStateChecked = True
  End With
  With objDynDialog.QualityCodeStateValues(1)
    .VALUE_BAD_COMMLUV = 20
    .VALUE_BAD_COMMNUV = 30
    .VALUE_BAD_CONFERROR = 40
    .VALUE_BAD_DEVICE = 60
    .VALUE_BAD_MISCSTATES = 70
    .VALUE_BAD_NONSPECIFIC = 80
    .VALUE_BAD_NOTCONNECTED = 90
    .VALUE_BAD_OUTOFSEAR = 100
    .VALUE_BAD_PROCRELNOM = 110
    .VALUE_BAD_PROCRELSUB = 120
    .VALUE_HIGHLIMITED = 130
    .VALUE_LOWLIMITED = 140
    .VALUE_UNCERT_ENGVHIGHLIM = 150
    .VALUE_UNCERT_ENGVLOWLIM = 160
    .VALUE_UNCERT_INITVAL = 170
    .VALUE_UNCERT_LUV = 180
    .VALUE_UNCERT_MAINTDEM = 190
    .VALUE_UNCERT_MISCSTATES = 200
    .VALUE_UNCERT_NONSPECIFIC = 210
    .VALUE_UNCERT_PROCRELNOM = 220
    .VALUE_UNCERT_SIMVAL = 230
    .VALUE_UNCERT_SUBSTSET = 240
  End With
End Sub
```

See also

VALUE_BAD_CONFERROR Property (Page 2478)

QualityCodeStateChecked Properties (Page 2390)
VALUE_BAD_PROCRELNOM Property

Description

Specifies the value assigned to a dynamized property if quality code "bad, process related, no maintenance" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic
dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a
quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMIcircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMIcircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

VALUE_UNCERT_LUV Property (Page 2530)

QualityCodeStateChecked Properties (Page 2390)
VALUE_BAD_PROCRELSUB Property

**Description**

Specifies the value assigned to a dynamized property if quality code "bad, process related, substitute value" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMIObject
Set objCircle = ActiveDocument.HMIObjs.AddHMIObject("Circle_A", "HMIObject")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'"
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  With objDynDialog.QualityCodeStateValues(1)
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End With
End Sub
```

See also

- VALUE_UNCERT_PROCRELNOM Property (Page 2538)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_CONVERSION_ERROR Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Conversion error" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA772
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        ,
        'Activate analysis of variablestate
        .VariableStateChecked = True
    End With
    With objDynDialog.VariableStateValues(1)
        ,
        'define a value for every state:
        .VALUE_ACCESS_FAULT = 20
        .VALUE_ADDRESS_ERROR = 30
        .VALUE_CONVERSION_ERROR = 40
        .VALUE_HANDSHAKE_ERROR = 60
        .VALUE_HARDWARE_ERROR = 70
        .VALUE_INVALID_KEY = 80
        .VALUE_MAX_LIMIT = 90
        .VALUE_MAX_RANGE = 100
        .VALUE_MIN_LIMIT = 110
        .VALUE_MIN_RANGE = 120
        .VALUE_NOT_ESTABLISHED = 130
        .VALUE_SERVERDOWN = 140
        .VALUE_STARTUP_VALUE = 150
        .VALUE_TIMEOUT = 160
    End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
**VALUE_HANDSHAKE_ERROR Property**

**Description**

Defines or returns the value that will be assigned to the dynamic property if tag status "Handshake error" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA773
Dim objDynDialog As HMI.DynamicDialog
Dim objCircle As HMI.Circle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMI.Circle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmi.DynamicCreationTypeDynamicDialog, "'NewDynamic1'")
With objDynDialog
  .ResultType = hmi.ResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  'Activate analysis of variablestate
  .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
  'define a value for every state:
  .VALUE_ACCESS_FAULT = 20
  .VALUE_ADDRESS_ERROR = 30
  .VALUE_CONVERSION_ERROR = 40
  .VALUE_HANDSHAKE_ERROR = 60
  .VALUE_HARDWARE_ERROR = 70
  .VALUE_INVALID_KEY = 80
  .VALUE_MAX_LIMIT = 90
  .VALUE_MAX_RANGE = 100
  .VALUE_MIN_LIMIT = 110
  .VALUE_MIN_RANGE = 120
  .VALUE_NOT_ESTABLISHED = 130
  .VALUE_SERVERDOWN = 140
  .VALUE_STARTUP_VALUE = 150
  .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
VALUE_HARDWARE_ERROR Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "No network module" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA774
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
  "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    'Activate analysis of variablestate
    .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    'define a value for every state:
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
End Sub
```

See also

- `VALUE_MAX_RANGE Property (Page 2508)`
- `VariableStateChecked Property (Page 2545)`
- `VALUE_TIMEOUT Property (Page 2520)`
- `VALUE_STARTUP_VALUE Property (Page 2518)`
- `VALUE_SERVERDOWN Property (Page 2516)`
- `VALUE_NOT_ESTABLISHED Property (Page 2514)`
- `VALUE_MIN_RANGE Property (Page 2512)`
VALUE_MIN_LIMIT Property (Page 2510)
VALUE_MAX_LIMIT Property (Page 2506)
VALUE_INVALID_KEY Property (Page 2502)
VALUE_HANDSHAKE_ERROR Property (Page 2496)
VALUE_CONVERSION_ERROR Property (Page 2494)
VALUE_ADDRESS_ERROR Property (Page 2472)
VALUE_ACCESS_FAULT Property (Page 2470)
VariableStateValue Object (Page 2096)

VALUE_HIGHLIMITED Property

Description

Specifies the value assigned to a dynamized property if quality code "high limited" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnallogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

- QualityCodeStateChecked Properties (Page 2390)
- VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMTED Property (Page 2504)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

VALUE_INVALID_KEY Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Tag not found" occurs.
The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA775
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
  "'NewDynamic1'"
  )
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  ',
  'Activate analysis of variable state
  .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
  ',
  'define a value for every state:
  .VALUE_ACCESS_FAULT = 20
  .VALUE_ADDRESS_ERROR = 30
  .VALUE_CONVERSION_ERROR = 40
  .VALUE_HANDSHAKE_ERROR = 60
  .VALUE_HARDWARE_ERROR = 70
  .VALUE_INVALID_KEY = 80
  .VALUE_MAX_LIMIT = 90
  .VALUE_MAX_RANGE = 100
  .VALUE_MIN_LIMIT = 110
  .VALUE_MIN_RANGE = 120
  .VALUE_NOT_ESTABLISHED = 130
  .VALUE_SERVERDOWN = 140
  .VALUE_STARTUP_VALUE = 150
  .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
VALUE_MAX_RANGE Property (Page 2508)
VALUE_MAX_LIMIT Property (Page 2506)
VALUE_HARDWARE_ERROR Property (Page 2498)
VALUE_HANDSHAKE_ERROR Property (Page 2496)
VALUE_CONVERSION_ERROR Property (Page 2494)
VALUE_ADDRESS_ERROR Property (Page 2472)
VALUE_ACCESS_FAULT Property (Page 2470)
VariableStateValue Object (Page 2096)

VALUE_LOWLIMITED Property

Description

Specifies the value assigned to a dynamized property if quality code "low limited" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjets.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
    End With
    With objDynDialog.QualityCodeStateValues(1)
        'define a value for every state:
        .VALUE_BAD_COMMUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

VALUE_BAD_PROCRELSUB Property (Page 2492)
QualityCodeStateChecked Properties (Page 2390)
VALUE_MAX_LIMIT Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Upper limit exceeded" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA776
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "NewDynamic1")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
End Sub

See also

- VALUE_MIN_LIMIT Property (Page 2510)
- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
VALUE_MAX_RANGE Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Format upper limit exceeded" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA777
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
VALUE_MIN_LIMIT Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Lower limit exceeded" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA778
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MAX_RANGE Property (Page 2508)
VALUE_MIN_RANGE Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Format lower limit exceeded" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA779
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  , 'Activate analysis of variablestate
  .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
  , 'Define a value for every state:
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_LIMIT Property (Page 2510)
- VALUE_MAX_RANGE Property (Page 2508)
**VALUE_MAX_LIMIT Property** (Page 2506)

**VALUE_INVALID_KEY Property** (Page 2502)

**VALUE_HARDWARE_ERROR Property** (Page 2498)

**VALUE_HANDSHAKE_ERROR Property** (Page 2496)

**VALUE_CONVERSION_ERROR Property** (Page 2494)

**VALUE_ADDRESS_ERROR Property** (Page 2472)

**VALUE_ACCESS_FAULT Property** (Page 2470)

**VariableStateValue Object** (Page 2096)

**VALUE_NOT_ESTABLISHED Property**

**Description**

Defines or returns the value that will be assigned to the dynamic property if tag status "No check-back message from the channel" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA780
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
    "'NewDynamic1'")
With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

VariableStateChecked Property (Page 2545)
VALUE_TIMEOUT Property (Page 2520)
VALUE_STARTUP_VALUE Property (Page 2518)
VALUE_SERVERDOWN Property (Page 2516)
VALUE_MIN_RANGE Property (Page 2512)
VALUE_MIN_LIMIT Property (Page 2510)
VALUE_MAX_RANGE Property (Page 2508)
VALUE_MAX_LIMIT Property (Page 2506)
VALUE_INVALID_KEY Property (Page 2502)
VALUE_HARDWARE_ERROR Property (Page 2498)
VALUE_HANDSHAKE_ERROR Property (Page 2496)
VALUE_CONVERSION_ERROR Property (Page 2494)
VALUE_ADDRESS_ERROR Property (Page 2472)
VALUE_ACCESS_FAULT Property (Page 2470)
VariableStateValue Object (Page 2096)

VALUE_SERVERDOWN Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "Server not available" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA781
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
  ' Activate analysis of variablenstate
    .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    'define a value for every state:
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
  End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
- VALUE_MAX_RANGE Property (Page 2508)
**VALUE_MAX_LIMIT Property** (Page 2506)

**VALUE_INVALID_KEY Property** (Page 2502)

**VALUE_HARDWARE_ERROR Property** (Page 2498)

**VALUE_HANDSHAKE_ERROR Property** (Page 2496)

**VALUE_CONVERSION_ERROR Property** (Page 2494)

**VALUE_ADDRESS_ERROR Property** (Page 2472)

**VALUE_ACCESS_FAULT Property** (Page 2470)

**VariableStateValue Object** (Page 2096)

---

**VALUE_STARTUP_VALUE Property**

**Description**

Defines or returns the value that will be assigned to the dynamic property if tag status "Start value" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA782
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .VariableStateChecked = True
    End With
    With objDynDialog.VariableStateValues(1)
        .VALUE_ACCESS_FAULT = 20
        .VALUE_ADDRESS_ERROR = 30
        .VALUE_CONVERSION_ERROR = 40
        .VALUE_HANDSHAKE_ERROR = 60
        .VALUE_HARDWARE_ERROR = 70
        .VALUE_INVALID_KEY = 80
        .VALUE_MAX_LIMIT = 90
        .VALUE_MAX_RANGE = 100
        .VALUE_MIN_LIMIT = 110
        .VALUE_MIN_RANGE = 120
        .VALUE_NOT_ESTABLISHED = 130
        .VALUE_SERVERDOWN = 140
        .VALUEStartup_VALUE = 150
        .VALUE_TIMEOUT = 160
    End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_TIMEOUT Property (Page 2520)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
- VALUE_MAX_RANGE Property (Page 2508)
VALUE_TIMEOUT Property

Description

Defines or returns the value that will be assigned to the dynamic property if tag status "No connection" occurs.

The value of the VariableStateChecked property must be TRUE in order for the status to be evaluated.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. The dynamization takes place by evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA783
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
    "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
  End With
  'Activate analysis of variablestate
  .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    'define a value for every state:
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
End Sub
```

See also

- VariableStateChecked Property (Page 2545)
- VALUE_STARTUP_VALUE Property (Page 2518)
- VALUE_SERVERDOWN Property (Page 2516)
- VALUE_NOT_ESTABLISHED Property (Page 2514)
- VALUE_MIN_RANGE Property (Page 2512)
- VALUE_MIN_LIMIT Property (Page 2510)
- VALUE_MAX_RANGE Property (Page 2508)
VALUE_MAX_LIMIT Property (Page 2506)
VALUE_INVALID_KEY Property (Page 2502)
VALUE_HARDWARE_ERROR Property (Page 2498)
VALUE_HANDSHAKE_ERROR Property (Page 2496)
VALUE_CONVERSION_ERROR Property (Page 2494)
VALUE_ADDRESS_ERROR Property (Page 2472)
VALUE_ACCESS_FAULT Property (Page 2470)
VariableStateValue Object (Page 2096)

VALUE_UNCERT_ENGVHIGHLIM Property

Description

Specifies the value assigned to a dynamized property if quality code "uncertain, engineering unit range violation, high limit set" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFserv = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

VALUE_BAD_PROCRELSUB Property (Page 2492)
QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

**VALUE_UNCERT_ENGVLOWLIM Property**

**Description**

Specifies the value assigned to a dynamized property if quality code "uncertain, engineering unit range violation, low limit set" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog_QUALITYCODESTATEVALUES(1)
  'define a value for every state:
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub

See also
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_LUV Property (Page 2530)
VALUE_UNCERT_INITVAL Property (Page 2528)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUNV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

VALUE_UNCERT_ENGVONLIM Property

Description

Specifies the value assigned to a dynamized property if quality code "uncertain, engineering unit range violation, on limits set" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
'"NewDynamic1"
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSE RV = 100
  .VALUE_BAD_PRCRELNOM = 110
  .VALUE_BAD_PRCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PRCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

- VALUE_UNCERT_SIMVAL Property (Page 2540)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_INITVAL Property

**Description**

Specifies a value assigned to a dynamized property if quality code "uncertain, initial value" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        'define a value for every state:
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSEv = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

VALUE_UNCERT_LUV Property (Page 2530)

QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property (Page 2542)
VALUE_UNCERT_SIMVAL Property (Page 2540)
VALUE_UNCERT_PROCRELNOM Property (Page 2538)
VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
VALUE_UNCERT_MISCSTATES Property (Page 2534)
VALUE_UNCERT_MAINTDEM Property (Page 2532)
VALUE_UNCERT_ENGVONLIM Property (Page 2526)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2524)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2522)
VALUE_LOWLIMITED Property (Page 2504)
VALUE_HIGHLIMITED Property (Page 2500)
VALUE_BAD_PROCRELSUB Property (Page 2492)
VALUE_BAD_PROCRELNOM Property (Page 2490)
VALUE_BAD_OUTOFSERV Property (Page 2488)
VALUE_BAD_NOTCONNECTED Property (Page 2486)
VALUE_BAD_NONSPECIFIC Property (Page 2484)
VALUE_BAD_MISCSTATES Property (Page 2482)
VALUE_BAD_DEVICE Property (Page 2480)
VALUE_BAD_CONFERROR Property (Page 2478)
VALUE_BAD_COMMNUV Property (Page 2476)
VALUE_BAD_COMMLUV Property (Page 2474)
QualityCodeStateValue Object (Page 2044)

VALUE_UNCERT_LUV Property

Description

Specifies a value assigned to a dynamized property if quality code "uncertain, last usable value" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
        "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITIVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

- VALUE_HIGHLIMITED Property (Page 2500)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_MAINTDEM Property

Description

Specifies a value assigned to a dynamized property if quality code "uncertain, maintenance demanded" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMI.circle
Set objCircle = ActiveDocument.HMIOBjects.AddHMIObject("Circle_A", "HMIcircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, ""NewDynamic1")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200

  'Activate analysis of qualitycodestate
  .QualityCodeStateChecked = True
End With
End Sub
```

See also

VALUE_UNCERT_NONSPECIFIC Property (Page 2536)
QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_MISCSTATES Property

Description

Specifies the value assigned to a dynamized property if quality code "uncertain miscellaneous states" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
With objDynDialog.QualityCodeStateValues(1)
    'define a value for every state:
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BADDEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

- VALUE_LOWLIMITED Property (Page 2504)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_NONSPECIFIC Property

Description

Specifies the value assigned to a dynamized property if quality code "uncertain, non-specific" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, 
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .QualityCodeStateChecked = True
End With
With objDynDialog.QualityCodeStateValues(1)
  'define a value for every state:
  .VALUE_BAD_COMMLUV = 20
  .VALUE_BAD_COMMNUV = 30
  .VALUE_BAD_CONFERROR = 40
  .VALUE_BAD_DEVICE = 60
  .VALUE_BAD_MISCSTATES = 70
  .VALUE_BAD_NONSPECIFIC = 80
  .VALUE_BAD_NOTCONNECTED = 90
  .VALUE_BAD_OUTOFSERV = 100
  .VALUE_BAD_PROCRELNOM = 110
  .VALUE_BAD_PROCRELSUB = 120
  .VALUE_HIGHLIMITED = 130
  .VALUE_LOWLIMITED = 140
  .VALUE_UNCERT_ENGVHIGHLIM = 150
  .VALUE_UNCERT_ENGVLOWLIM = 160
  .VALUE_UNCERT_INITVAL = 170
  .VALUE_UNCERT_LUV = 180
  .VALUE_UNCERT_MAINTDEM = 190
  .VALUE_UNCERT_MISCSTATES = 200
  .VALUE_UNCERT_NONSPECIFIC = 210
  .VALUE_UNCERT_PROCRELNOM = 220
  .VALUE_UNCERT_SIMVAL = 230
  .VALUE_UNCERT_SUBSTSET = 240
End With
End Sub
```

See also

- VALUE_UNCERT_MAINTDEM Property (Page 2532)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_PROCRELNOM Property

Description

Specifies the value assigned to a dynamized property if quality code "uncertain, process related, no maintenance" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
    'VBA770
    Dim objDynDialog As HMIDynamicDialog
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
    Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog, "'NewDynamic1'")
    With objDynDialog
        .ResultType = hmiResultTypeAnalog
        .AnalogResultInfos.ElseCase = 200
        .QualityCodeStateChecked = True
    End With
    With objDynDialog.QualityCodeStateValues(1)
        .VALUE_BAD_COMMLUV = 20
        .VALUE_BAD_COMMNUV = 30
        .VALUE_BAD_CONFERROR = 40
        .VALUE_BAD_DEVICE = 60
        .VALUE_BAD_MISCSTATES = 70
        .VALUE_BAD_NONSPECIFIC = 80
        .VALUE_BAD_NOTCONNECTED = 90
        .VALUE_BAD_OUTOFSERV = 100
        .VALUE_BAD_PROCRELNOM = 110
        .VALUE_BAD_PROCRELSUB = 120
        .VALUE_HIGHLIMITED = 130
        .VALUE_LOWLIMITED = 140
        .VALUE_UNCERT_ENGVHIGHLIM = 150
        .VALUE_UNCERT_ENGVLOWLIM = 160
        .VALUE_UNCERT_INITVAL = 170
        .VALUE_UNCERT_LUV = 180
        .VALUE_UNCERT_MAINTDEM = 190
        .VALUE_UNCERT_MISCSTATES = 200
        .VALUE_UNCERT_NONSPECIFIC = 210
        .VALUE_UNCERT_PROCRELNOM = 220
        .VALUE_UNCERT_SIMVAL = 230
        .VALUE_UNCERT_SUBSTSET = 240
    End With
End Sub
```

See also

- VALUE_BAD_COMMNUV Property (Page 2476)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SIMVAL Property

Description

Specifies a value assigned to a dynamized property if quality code "uncertain, simulated value" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA770
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
End With
End Sub
```

See also

- VALUE_BAD_PROCRELNOM Property (Page 2490)
- QualityCodeStateChecked Properties (Page 2390)
VALUE_UNCERT_SUBSTSET Property

**Description**

Specifies a value assigned to a dynamized property if quality code "uncertain, substitute set" occurs, or returns its value.

In order for the quality code to be analyzed, the QualityCodeStateChecked property must be TRUE.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic dialog. Dynamization uses the analysis of the quality code of a tag. If the tag fails to return a quality code, a substitute value (ElseCase property) is defined:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA770
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
                          "'NewDynamic1'")
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
  End With
  With objDynDialog.QualityCodeStateValues(1)
    .VALUE_BAD_COMMLUV = 20
    .VALUE_BAD_COMMNUV = 30
    .VALUE_BAD_CONFERROR = 40
    .VALUE_BAD_DEVICE = 60
    .VALUE_BAD_MISCSTATES = 70
    .VALUE_BAD_NONSPECIFIC = 80
    .VALUE_BAD_NOTCONNECTED = 90
    .VALUE_BAD_OUTOFSERV = 100
    .VALUE_BAD_PROCRELNOM = 110
    .VALUE_BAD_PROCRELSUB = 120
    .VALUE_HIGHLIMITED = 130
    .VALUE_LOWLIMITED = 140
    .VALUE_UNCERT_ENGVHIGHLIM = 150
    .VALUE_UNCERT_ENGVLOWLIM = 160
    .VALUE_UNCERT_INITVAL = 170
    .VALUE_UNCERT_LUV = 180
    .VALUE_UNCERT_MAINTDEM = 190
    .VALUE_UNCERT_MISCSTATES = 200
    .VALUE_UNCERT_NONSPECIFIC = 210
    .VALUE_UNCERT_PROCRELNOM = 220
    .VALUE_UNCERT_SIMVAL = 230
    .VALUE_UNCERT_SUBSTSET = 240
  End With
End Sub
```

See also

- VALUE_UNCERT_MISCSTATES Property (Page 2534)
- QualityCodeStateChecked Properties (Page 2390)
VariablesExist Property

Description

TRUE when all the tags used in the source code of a DynamicDialog object are defined. Read only access.

You can use this property to check whether all the tags that you have defined in the source code of the Dynamic dialog are created in WinCC.

Example:

--

See also

DynamicDialog Object (Page 1955)
VariableStateChecked Property

Description

TRUE if the status of the specified tag is used in the dynamic dialog for dynamization.
BOOLEAN write-read access.

Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic
dialog. The dynamization takes place by evaluating the status of a tag. If a tag does not return
a status, a substitute value (ElseCase property) is defined, a tag name is issued and three
analog value ranges are created:

```vba
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA785
Dim objDynDialog As HMIDynamicDialog
Dim objCircle As HMIObject
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMIObject")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
    "'NewDynamic1'")
With objDynDialog
  .ResultType = hmiResultTypeAnalog
  .AnalogResultInfos.ElseCase = 200
  .VariableStateChecked = True
End With
With objDynDialog.VariableStateValues(1)
  'define a value for every state:
  .VALUE_ACCESS_FAULT = 20
  .VALUE_ADDRESS_ERROR = 30
  .VALUE_CONVERSION_ERROR = 40
  .VALUE_HANDSHAKE_ERROR = 60
  .VALUE_HARDWARE_ERROR = 70
  .VALUE_INVALID_KEY = 80
  .VALUE_MAX_LIMIT = 90
  .VALUE_MAX_RANGE = 100
  .VALUE_MIN_LIMIT = 110
  .VALUE_MIN_RANGE = 120
  .VALUE_NOT_ESTABLISHED = 130
  .VALUE_SERVERDOWN = 140
  .VALUE_STARTUP_VALUE = 150
  .VALUE_TIMEOUT = 160
End With
End Sub
```

See also

DynamicDialog Object (Page 1955)
VariableStateType Property

Description

Returns the type of tag monitoring used to dynamize a property or an event: No monitoring, quality code, or tag status. Read only access.

<table>
<thead>
<tr>
<th>Index</th>
<th>VariableStateType</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hmiNoVariableState</td>
</tr>
<tr>
<td>1</td>
<td>hmiVariableQCState</td>
</tr>
<tr>
<td>2</td>
<td>hmiVariableState</td>
</tr>
</tbody>
</table>

Example:

The procedure "GetVariableStateType()" reads the type of monitoring from the current document. In this example, the type of monitoring is output in a message:

```vba
Sub GetVariableStateType()
'VBA819
Dim objDyn As HMIDynamicDialog
Set objDyn =
ActiveDocument.Properties("Width").CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'TestVal'")
MsgBox objDyn.VariableStateType
objDyn.Delete
End Sub
```

See also

DynamicDialog Object (Page 1955)

VariableStateValues Property

Description

Returns the VariableStateValues listing. Use the VariableStateValues property with the Item property to assign a value to the tag status to be used for dynamization.
Example:

In the following example the radius of a circle will be dynamically configured using the Dynamic
dialog. The dynamization takes place by evaluating the status of a tag. If a tag does not return
a status, a substitute value (ElseCase property) is defined, a tag name is issued and three
analog value ranges are created:

Sub AddDynamicDialogToCircleRadiusTypeAnalog()
  'VBA786
  Dim objDynDialog As HMIDynamicDialog
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
  Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
    "'NewDynamic1'");
  With objDynDialog
    .ResultType = hmiResultTypeAnalog
    .AnalogResultInfos.ElseCase = 200
    .VariableStateChecked = True
  End With
  With objDynDialog.VariableStateValues(1)
    .VALUE_ACCESS_FAULT = 20
    .VALUE_ADDRESS_ERROR = 30
    .VALUE_CONVERSION_ERROR = 40
    .VALUE_HANDSHAKE_ERROR = 60
    .VALUE_HARDWARE_ERROR = 70
    .VALUE_INVALID_KEY = 80
    .VALUE_MAX_LIMIT = 90
    .VALUE_MAX_RANGE = 100
    .VALUE_MIN_LIMIT = 110
    .VALUE_MIN_RANGE = 120
    .VALUE_NOT_ESTABLISHED = 130
    .VALUE_SERVERDOWN = 140
    .VALUE_STARTUP_VALUE = 150
    .VALUE_TIMEOUT = 160
  End With
End Sub

See also

VariableStateValues Object (Listing) (Page 2097)
DynamicDialog Object (Page 1955)
VariableTriggers Property

Description

Returns the VariableTriggers listing. Use the VariableTriggers property in order to add a tag trigger to a VB action or C action.

Example:

In the following example the radius of a circle is made dynamic with the aid of a VB script. A tag trigger is used as the trigger:

```vba
Sub DynamicWithVariableTrigger()
  'VBA787
  Dim objVBScript As HMIScriptInfo
  Dim objVarTrigger As HMIVariableTrigger
  Dim objCircle As HMICircle
  Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle_VariableTrigger", "HMICircle")
  Set objVBScript = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeVBScript)
  With objVBScript
    'Triggername and cycletime are defined by add-methode
    Set objVarTrigger = .Trigger.VariableTriggers.Add("VarTrigger", hmiVariableCycleType_10s)
    .SourceCode = ""
  End With
End Sub
```

See also

VariableTriggers Object (Listing) (Page 2101)

VarName Property

Description

Defines the tag whose status is to be used in the Dynamic dialog for the purpose of dynamics, or returns the name.
Example:

In this example the name of the trigger tag used for creating dynamics in the radius of a circle will be output:

```vba
Sub GetVarName()
    Dim objVBScript As HMIScriptInfo
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.Item("Circle_VariableTrigger")
    Set objVBScript = objCircle.Radius.Dynamic
    With objVBScript
        MsgBox "The radius is dynamicabled with: " & .Trigger.VariableTriggers.Item(1).VarName
    End With
End Sub
```

See also

VariableStateValue Object (Page 2096)

VBAVersion Property

Description

Returns the VBA version number. Read only access.

Example:

In the following example the current VBA version number is output:

```vba
Sub ShowVBAVersion()
    MsgBox Application.VBAVersion
End Sub
```

See also

Application Object (Page 1909)
VBE Property

Description

Returns the VB Extensibility object. Read access.

Example:

See also

Application Object (Page 1909)

Version Property

Description

Returns the version number of the specified application. Read only access.

Example:

In the following example the version number of the Graphics Designer is output:

```vba
Sub ShowVersionOfGraphicsDesigner()
    'VBA791
    MsgBox Application.Version
End Sub
```

See also

Application Object (Page 1909)

Views Property

Description

Returns the Views listing. Use the Views listing to create a new copy of a picture, for instance.
Example:
In the following example a copy of the active picture is created and then activated:

Sub AddView()
    'VBA792
    Dim objView As HMIView
    Set objView = ActiveDocument.Views.Add
    objView.Activate
End Sub

See also
Views Object (Listing) (Page 2103)

Visible Property

Description
TRUE if the specified object is intended to be visible. BOOLEAN write-read access.

Example:
In the following example a circle will be inserted into the active picture. This circle is not intended to be visible in Runtime:

Sub HideCircleInRuntime()
    'VBA793
    Dim objCircle As HMICircle
    Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMICircle")
    objCircle.Visible = False
End Sub

See also
Toolbartem Object (Page 2083)
Menultem Object (Page 2013)
HMIObject Object (Page 1988)
Document Object (Page 1949)
Toolbar Object (Page 2080)
Menu Object (Page 2009)
Application Object (Page 1909)
WarningHigh Property

Description
Defines or returns the high limit value "Warning High" in the case of the BarGraph object. The "CheckWarningHigh" property must be set to "True" in order for the limit value to be monitored. The display on reaching the limit value and the type of evaluation are defined via the properties ColorWarningHigh and TypeWarningHigh.

Example:
The "BarGraphLimitConfiguration()" procedure configures the properties of the high limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "75".

```vba
Sub BarGraphLimitConfiguration()
    'VBA794
    Dim objBarGraph As HMIBarGraph
    Set objBarGraph = ActiveDocument.HMIObjcts.AddHMIObjct("Bar1", "HMIBarGraph")
    With objBarGraph
        'Set analysis = absolute
        .TypeWarningHigh = False
        'Activate monitoring
        .CheckWarningHigh = True
        'Set barcolor = "red"
        .ColorWarningHigh = RGB(255, 0, 0)
        'Set upper limit = "75"
        .WarningHigh = 75
    End With
End Sub
```

See also
TypeWarningHigh Property (Page 2458)
ColorWarningHigh Property (Page 2192)
CheckWarningHigh Property (Page 2178)
BarGraph Object (Page 1914)
WarningLow Property

Description

Defines or returns the low limit value "Warning Low" in the case of the BarGraph object. The "CheckWarningLow" property must be set to "True" in order for the limit value to be monitored.

The display on reaching the limit value and the type of evaluation are defined via the properties ColorWarningLow and TypeWarningLow.

Example:

The "BarGraphLimitConfiguration()" procedure configures the properties of the low limit value for an alarm. In this example the type of evaluation will be set to "Absolute". The alarm will be triggered at a value of "12".

```vba
Sub BarGraphLimitConfiguration()
  'VBA795
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    'Set analysis = absolute
    .TypeWarningLow = False
    'Activate monitoring
    .CheckWarningLow = True
    'Set barcolor = "magenta"
    .ColorWarningLow = RGB(255, 0, 255)
    'Set lower limit = "12"
    .WarningLow = 75
  End With
End Sub
```

See also

TypeWarningLow Property (Page 2459)
ColorWarningLow Property (Page 2193)
CheckWarningLow Property (Page 2179)
BarGraph Object (Page 1914)

Width Property

Description

Defines or returns the width of an object in pixels.
Example:

In the following example three objects of different sizes will be inserted in the active picture. Then all objects will be selected and set to the same width:

Sub ApplySameWidthToSelectedObjects()
'VBA796
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objEllipse As HMIEllipse
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("sCircle", "HMICircle")
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("sRectangle", "HMIRectangle")
Set objEllipse = ActiveDocument.HMIObjects.AddHMIObject("sEllipse", "HMIEllipse")
With objCircle
    .Top = 30
    .Left = 0
    .Width = 15
    .Selected = True
End With
With objRectangle
    .Top = 80
    .Left = 42
    .Width = 40
    .Selected = True
End With
With objEllipse
    .Top = 48
    .Left = 162
    .Width = 120
    .BackColor = RGB(255, 0, 0)
    .Selected = True
End With
MsgBox "Objects selected!"
ActiveDocument.Selection.SameWidth
End Sub

See also

HMIOBJECT Object (Page 1988)

WinCCStyle property

Description

Defines the style in which the object is displayed.

User-defined Shows the object according to the respective settings.
global Shows the object in a globally defined design.
Windows Style Shows the object in Windows style.
Example

WindowBorder Property

Description
TRUE if it is intended that the application window or picture window shall be displayed with a border in Runtime. BOOLEAN write-read access.

Example:
The "ApplicationWindowConfig" procedure accesses the properties of the application window. In this example the application window will

```
Sub ApplicationWindowConfig()
  'VBA797
  Dim objAppWindow As HMIApplicationWindow
  Set objAppWindow = ActiveDocument.HMIOObjects.AddHMIOObject("AppWindow", "HMIApplicationWindow")
  With objAppWindow
    .Caption = True
    .CloseButton = False
    .Height = 200
    .Left = 10
    .MaximizeButton = True
    .Moveable = False
    .OnTop = True
    .Sizeable = True
    .Top = 20
    .Visible = True
    .Width = 250
    .WindowBorder = True
  End With
End Sub
```

See also
PictureWindow Object (Page 2029)
ApplicationWindow Object (Page 1911)
**WindowMonitorNumber property**

**Description**

Defines the monitor on which the picture window is displayed. This requires that the system supports more than one monitor. The attribute is only effective if the "Independent window" attribute is set to "Yes".

1-n The number of the monitor in the operating system on which the picture window is displayed.

**Example**

**WindowPositionMode property**

**Description**

Defines the position and scaling of the picture window on the screen. The property is only effective if the "Independent window" attribute is set to "Yes".

- **Standard** The picture window is positioned in its original size in the configured position on the screen.
- **Center** The picture window is positioned in its original size, centered on the screen.
- **Maximize** The picture window is scaled to the size of the screen.

**Example**
WindowsStyle property

Description
Defines whether the object is displayed in the Windows style of WinCC version 6.2. It can only be selected if "WinCC Classic" is chosen as the current design.

yes Shows the object using the Windows style from WinCC version 6.2.
No Shows the object not using the Windows style from WinCC version 6.2.

Example

WindowState Property

Description
Returns the status of the window containing the specified application. READ access.

<table>
<thead>
<tr>
<th>WindowState</th>
<th>Assigned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximized</td>
<td>0</td>
</tr>
<tr>
<td>Minimized</td>
<td>1</td>
</tr>
<tr>
<td>Custom sized</td>
<td>2</td>
</tr>
</tbody>
</table>

Example:
In the following example the window status of the Graphics Designer is output:

Sub ShowWindowState()
"VBA798"
Dim strState As String
Select Case Application.WindowsState
Case 0
strState = "The application-window is maximized"
Case 1
strState = "The applicationwindow is minimized"
Case 2
strState = "The application-window has a userdefined size"
End Select
MsgBox strState
End Sub
See also

Application Object (Page 1909)

ZeroPoint Property

Description

Defines or returns the position of the zero point on the bar in the case of the BarGraph object. Specify the value as a %age of the total bar height. The zero point can also be outside of the range represented.

The "ScalingType" property must be set to "2" and "Scaling" must be set to "True".

Example:

The "BarGraphConfiguration()" procedure configures In this example the zero point is located halfway up the bar height:

```
Sub BarGraphConfiguration()
  'VBA799
  Dim objBarGraph As HMIBarGraph
  Set objBarGraph = ActiveDocument.HMIObjets.AddHMIObject("Bar1", "HMIBarGraph")
  With objBarGraph
    .Scaling = True
    .ScalingType = 2
    .ZeroPoint = 50
    .ZeroPointValue = 0
  End With
End Sub
```

See also

ZeroPointValue Property (Page 2559)
ScalingType Property (Page 2406)
Scaling Property (Page 2405)
BarGraph Object (Page 1914)
ZeroPointValue Property

Description

Defines or returns the absolute value for the zero point.

Example:

The “BarGraphConfiguration()” procedure configures In this example the absolute value of the zero point will be set to "0".

```vba
Sub BarGraphConfiguration()
'VBA800
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
 .Scaling = True
 .ScalingType = 2
 .ZeroPointValue = 0
End With
End Sub
```

See also

ZeroPoint Property (Page 2558)
ScalingType Property (Page 2406)
Scaling Property (Page 2405)
BarGraph Object (Page 1914)
3DBarGraph Object (Page 1894)

Zoom Property

Description

Defines or returns the zoom factor.
Example:

In this example a copy of the active picture is created and the zoom factor is set to 50%:

```vba
Sub CreateViewFromActiveDocument()
'VBA801
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Zoom = 50
End Sub
```

See also

View Object (Page 2102)
PictureWindow Object (Page 2029)

3.6.2 VBA in Other WinCC Editors

3.6.2.1 VBA in Other WinCC Editors

Introduction

In addition to Graphics Designer, you can automate the following WinCC editors with VBA:

- Tag Management
- Tag Logging
- Text library
- Alarm Logging

The functions for accessing the editors are contained in the "HMIGO" class.

Requirement

- The "HMIGenObjects.dll" file is referenced. This happens automatically during WinCC installation.

Principle

For access to the "HMIGO" class with VBA, you must reference the "HMI GeneralObjects 1.0 Type Library" in the VBA editor via the "Tools > References" menu. You must create a new instance of this class in the program code, e.g.:

```vba
'Dim HMIGOObject As New HMIGO
```
Create several different objects of this class if access several objects at the same time. Two instances of the "HMIGO" class are required, for example, in Tag Logging: The first instance is required for access to the archive tags, the second instance for access to the process value archive.

Application

To enable you to use the functions and properties of the editors in VBA, you must have opened a project in WinCC.

You can then, for example, do the following directly from the program code:

- Create several tags and change the values
- Edit text entries in the TextLibrary
- Adapt messages.

Querying Object State

The "HMIGO" class has the enumeration "HMIGO_OBJECT_STATE" which returns the state of the specified object. The enumeration can return the following values:

- OBJECT_EMPTY (2): Connection to the object is not available.
- OBJECT_OPENED (3): Connection to objects exists. You can change and read its parameters.
- OBJECT_MODIFIED (4): An object's parameters have been changed. If the corresponding Commit function is not called, the changes are not saved.
- WINCC_CONNECTED (1): The object is connected to the WinCC project. By default this connection is established when a function is called the first time. To release the connection, use the instruction "HMIGO = nothing", for example.

Error Handling

Errors can occur when you use the "HMIGO" class. Use the "OnError" statement to respond to these error messages. The "OnError" statement must come before the call of a function from the HMIGeneralObjects class:

```vba
Sub CreateTag()
'HMIGO_000
Dim hmiGOTag as New HMIGO
On Error GoTo ErrorHandlerHMIGO
hmiGOTag.CreateTag "NewTag", TAG_BINARY_TAG, "ExistingConnection", "DB1,DD0,QC", "NewOrExistingGroupName"
...'...
Exit Sub
ErrorHandlerHMIGO:
MsgBox ("Error: " & Err.Number & " " & Err.Description & " " & Err.Source)
Resume Next
End Sub
```

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
As a result, an error text returned by the interface is output.

See also

VBA in Alarm Logging (Page 2622)
VBA in the Text Library (Page 2608)
VBA in Tag Logging (Page 2573)
VBA in Tag Management (Page 2562)

3.6.2.2 VBA in Tag Management

VBA in Tag Management

Introduction

VBA can be used to:

- Create tags directly from the program code
- Modify and delete tags
- Read out and change the properties of the tags
- Read out and change the types of the tags
- Read out and change the values of the tags

Note

The tags may not be open or opened in tag management when editing with VBA. If you wish to change the data type of a tag, you must first delete the tag and then regenerate it. You must save the parameters first in order to be able to transfer them following the generation of tags.

Principle

When you have created the instance of the “HMIGO” class, the following functions are available to you to access the tag management facility:

- CloseTag
- CommitTag
- CreateTag
- DeleteTag
- GetTag
- ListTag
The following enumerations are available for the parameter supply of these functions:

- HMIGO_TAG_TYPE
- HMIGO_TAG_LIST_TYPE

**Note**

If you set the start value to a binary tag, use the values "0" or "1". Do not use the values "False" or "True". These values are no longer valid for VBA programming in WinCC and will result in an error message.

Replace the values "False" and "True" with "0" and "1" in your existing VBA code.

**Access to the Object Properties**

You can also access the parameters of the above-mentioned functions directly in VBA by means of the following object properties:

<table>
<thead>
<tr>
<th>Object property</th>
<th>Description</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectStateTag</td>
<td>Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under &quot;VBA in other WinCC Editors&quot;.</td>
<td>Yes/no</td>
</tr>
<tr>
<td>TagName</td>
<td>Name of the tag</td>
<td>Yes/no</td>
</tr>
<tr>
<td>TagGroupName</td>
<td>Name of a group in which the tag is inserted. If the group does not yet exist, it is created. If no group name is specified, the tag is created outside all groups.</td>
<td>Yes/no</td>
</tr>
<tr>
<td>TagConnection</td>
<td>Name of a connection in which the tag and/or group is to be created. The connection must already be in existence, otherwise a tag cannot be created. If the name is omitted, an internal tag is created.</td>
<td>Yes/no</td>
</tr>
<tr>
<td>TagMaximum</td>
<td>Sets the new value of the upper limit</td>
<td>Yes/yes</td>
</tr>
<tr>
<td>TagMinimum</td>
<td>Sets the new value of the lower limit</td>
<td>Yes/yes</td>
</tr>
<tr>
<td>TagStart</td>
<td>Sets the new start value</td>
<td>Yes/yes</td>
</tr>
<tr>
<td>TagS5S7Address</td>
<td>Address of the S7 or S5 PLC to which the tag is connected. If no address is specified, a blank entry is passed.</td>
<td>Yes/yes</td>
</tr>
<tr>
<td>Object property</td>
<td>Description</td>
<td>Read/Write</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| TagType (HMI-GO_TAG_TYPE) | Data type of the tag. The possible types are:  
- TAG_BINARY_TAG (1)  
- TAG_SIGNED_8BIT_VALUE (2)  
- TAG_UNSIGNED_8BIT_VALUE (3)  
- TAG_SIGNED_16BIT_VALUE (4)  
- TAG_UNSIGNED_16BIT_VALUE (5)  
- TAG_SIGNED_32BIT_VALUE (6)  
- TAG_UNSIGNED_32BIT_VALUE (7)  
- TAG_FLOATINGPOINT_NUMBER_32BIT_IEEE_754 (8)  
- TAG_FLOATINGPOINT_NUMBER_64BIT_IEEE_754 (9)  
- TAG_TEXT_TAG_8BIT_CHARACTER_SET (10)  
- TAG_TEXT_TAG_16BIT_CHARACTER_SET (11)  
- TAG_RAW_DATA_TYPE (12)  
- TAG_STRUCT (14)  
- TAG_TEXT_REFERENCE (18) | Yes/no |
| TagUpdate (HMIGO_TAG_UPDATE) | Defines whether the tag is updated on the local computer or for the entire project. (for internal tag only)  
- TAG_COMPUTER_LOCAL (1)  
- TAG_PROJECT_WIDE (2) | Yes/yes |
| LengthText | Length of a text tag (0...255)  
"LengthText" can also be used for the length of the raw data tag. A testing of the correctness of the length will not be conducted. Observe the instructions of the communication channels. | yes/yes (only for external tag of type text) |
| TagScaleValid | Defines a linear scaling. | Yes/yes |
| TagScaleParam1 | Sets the value1 for the value range process. | Yes/yes |
| TagScaleParam2 | Sets the value2 for the value range process. | Yes/yes |
| TagScaleParam3 | Sets the value1 for the value range tag. | Yes/yes |
| TagScaleParam4 | Sets the value2 for the value range tag. | Yes/yes |
| TagStartvaluePersistence | Defines whether an internal tag is set as persistent. | Yes/yes |
| TagSubst | Replacement value (only for external variables) | Yes/yes |
| UseSubstValueOn-CommonError | Set the replacement value for connection errors. | Yes/yes |
| UseSubstValueOn-MaxLimit | Set the replacement value for upper limit. | Yes/yes |
| UseSubstValueOn-MinLimit | Set the replacement value for lower limit. | Yes/yes |
| UseSubstValueOn-StartValue | Set the replacement value for the start value. | Yes/yes |
You will find a description of the properties under the parameter descriptions for the corresponding functions.

**Note**
The "Tag synchronization" point in the property dialog of tags is not addressable with VBA. Tag synchronization is only available for internal tags.

For external tags, the "Type Conversion" point is not addressable with VBA.

---

**See also**

- ListTag function (Page 2572)
- GetTag Function (Page 2571)
- DeleteTag Function (Page 2570)
- CreateTag Function (Page 2568)
- CommitTag Function (Page 2566)
- CloseTag Function (Page 2565)
- VBA in Other WinCC Editors (Page 2560)

**CloseTag Function**

**Description**

Closes the open tag.

**Note**

Modified parameters are not saved.

**Syntax**

```
Expression.CloseTag()
```

**Expression**

Necessary. An expression which returns a "HMIGO" type object.

**Parameters**

--
Example:

Sub CloseTag()
    ' HMIGO_001
    ' procedure to close a variable
    ' tag need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strVariableName As String
    Set objHMIGO = New HMIGO
    strVariableName = "NewVariable"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'open a tag
    objHMIGO.GetTag strVariableName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'open a tag
    objHMIGO.CloseTag
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"

    Set objHMIGO = Nothing
End Sub

See also

ListTag function (Page 2572)
GetTag Function (Page 2571)
DeleteTag Function (Page 2570)
CreateTag Function (Page 2568)
CommitTag Function (Page 2566)
VBA in Tag Management (Page 2562)

CommitTag Function

Description

Writes the changed parameters of the open tag to WinCC.

Note

If further parameters are changed after a CommitTag call, write the changes to WinCC by calling this function again.
syntax

Expression.CommitTag()

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

--

Example:

Sub CommitTag()
    'HMIGO_002
    'procedure to change a property of a variable
    'tag need to be created before
dimensions
    Dim objHMIGO As HMIGO
    Dim strVariableName As String
    Set objHMIGO = New HMIGO
    strVariableName = "NewVariable"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'open a tag
    objHMIGO.GetTag strVariableName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'change a property
    objHMIGO.TagStart = 10
    'current status is "MODIFIED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'commit a tag
    objHMIGO.CommitTag
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"

    Set objHMIGO = Nothing
End Sub

See also

ListTag function (Page 2572)
GetTag Function (Page 2571)
DeleteTag Function (Page 2570)
CreateTag Function (Page 2568)
CreateTag Function

Description
Creates a new tag.

Syntax

Expression.CreateTag(TagName,TagType,[Connection],[S7S5Address],[GroupName])

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagName (string)</td>
<td>Name of the tag to be created.</td>
</tr>
<tr>
<td>TagType (HMIGO_TAG_TYPE)</td>
<td>Data type of the tag. The possible types are:</td>
</tr>
<tr>
<td></td>
<td>TAG_BINARY_TAG</td>
</tr>
<tr>
<td></td>
<td>TAG_SIGNED_8BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_UNSIGNED_8BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_SIGNED_16BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_UNSIGNED_16BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_SIGNED_32BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_UNSIGNED_32BIT_VALUE</td>
</tr>
<tr>
<td></td>
<td>TAG_FLOATINGPOINT_NUMBER_32BIT_IEEE_754</td>
</tr>
<tr>
<td></td>
<td>TAG_FLOATINGPOINT_NUMBER_64BIT_IEEE_754</td>
</tr>
<tr>
<td></td>
<td>TAG_TEXT_TAG_8BIT_CHARACTER_SET</td>
</tr>
<tr>
<td></td>
<td>TAG_TEXT_TAG_16BIT_CHARACTER_SET</td>
</tr>
<tr>
<td></td>
<td>TAG_RAW_DATA_TYPE</td>
</tr>
<tr>
<td></td>
<td>TAG_TEXT_REFERENCE</td>
</tr>
<tr>
<td>Connection (String, optional)</td>
<td>Name of a connection in which the tag and/or group is to be created. The connection must already be in existence, otherwise a tag cannot be created. If the name is omitted, an internal tag and/or group is recreated.</td>
</tr>
</tbody>
</table>
**Parameter (Data Type)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7S5Address (String, optional)</td>
<td>Address of the S7 or S5 PLC to which the tag is connected. Without an address indication, an empty entry will be handed over. The parameter “S7S5Address” must be supplemented by the string “,QC” for the configuration of the Quality Code, for example: “DB1,DD0,QC”. If the Quality Code of the tag is no longer to be monitored, the string “,QC” must be deleted.</td>
</tr>
<tr>
<td>GroupName (String, optional)</td>
<td>Name of a group in which the tag is inserted. If the group does not exist, it will be newly created. If the group name is not indicated, the tag will be created outside all groups.</td>
</tr>
</tbody>
</table>

**Example:**

```vba
Sub CreateTag()
    ' HMIGO_003
    ' procedure to create a variable
    ' tag must not be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strVariableName As String
    Set objHMIGO = New HMIGO
    strVariableName = "NewVariable"
    ' current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    ' create a tag
    objHMIGO.CreateTag strVariableName, TAG_SIGNED_32BIT_VALUE
    ' current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    Set objHMIGO = Nothing
End Sub
```

**See also**

- ListTag function (Page 2572)
- GetTag Function (Page 2571)
- DeleteTag Function (Page 2570)
- CommitTag Function (Page 2566)
- CloseTag Function (Page 2565)
- VBA in Tag Management (Page 2562)
DeleteTag Function

Description

Deletes the specified tag.

Syntax

Expression.DeleteTag(TagName)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagName (string)</td>
<td>Name of the tag to be deleted.</td>
</tr>
</tbody>
</table>

Example:

Sub DeleteTag()
    ' HMIGO_004
    ' procedure to delete a variable
    ' tag need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strVariableName As String
    Set objHMIGO = New HMIGO
    strVariableName = "NewVariable"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"

    'delete a tag
    objHMIGO.DeleteTag strVariableName
    Set objHMIGO = Nothing
End Sub

See also

ListTag function (Page 2572)
GetTag Function (Page 2571)
CreateTag Function (Page 2568)
CommitTag Function (Page 2566)
GetTag Function

Description

Reads in the parameters of the specified tag.
You can change or read the parameters by means of the object properties. You will find a list
of the available object properties in this documentation under "VBA in Tag Management".

Syntax

Expression.GetTag(TagName)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagName (string)</td>
<td>Name of the tag whose values are to be read in.</td>
</tr>
</tbody>
</table>

Example:

Sub GetTag()
' HMIGO_005
' procedure to open a variable
' tag need to be created before
' declarations
    Dim objHMIGO As HMIGO
    Dim strVariableName As String
    Set objHMIGO = New HMIGO
    strVariableName = "NewVariable"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    'open/ get a tag
    objHMIGO.GetTag strVariableName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTag, vbOKOnly, "Status Variable"
    Set objHMIGO = Nothing
End Sub
See also

ListTag function (Page 2572)
DeleteTag Function (Page 2570)
CreateTag Function (Page 2568)
CommitTag Function (Page 2566)
CloseTag Function (Page 2565)
VBA in Tag Management (Page 2562)

ListTag function

Description

Alternatively, the ListTag function returns the following contents of the Tag Management as a list:

- All the channel units created
- All the channels created
- All the connections created
- All the tag groups created
- All the tags created

Syntax

Expression.ListTag(ListType,pListArray,[Filter])

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListType (HMIGO_TAG_LIST_TYPE)</td>
<td>Defines which content should be returned as a list. Possibilities are:</td>
</tr>
<tr>
<td></td>
<td>- TAG_CHANNEL_UNITS (0) all channel units created</td>
</tr>
<tr>
<td></td>
<td>- TAG_CHANNELS (2) all channels created</td>
</tr>
<tr>
<td></td>
<td>- TAG_CONNECTIONS (3) all connections created</td>
</tr>
<tr>
<td></td>
<td>- TAG_GROUPS (4) all tag groups created</td>
</tr>
<tr>
<td></td>
<td>- TAG_NAMES (5) all tags created</td>
</tr>
<tr>
<td>pListArray (Variant)</td>
<td>List with the requested content.</td>
</tr>
<tr>
<td>Filter (String)</td>
<td>Filters can be set optionally. Wildcards &quot;***&quot; and &quot;?&quot; are also possible.</td>
</tr>
</tbody>
</table>
Example:

In the following example, a check is made whether the list with the connections created is empty because no connections have been set up:

```vba
Sub ReadTagByConnection()
  'HMIGO_027
  'read content in data manager by connections
  'no connections are implemented
  Dim objHMIGO As New HMIGO
  Dim varRange As Variant
  'read all connections
  objHMIGO.ListTag TAG_CONNECTIONS, arrContent
  'check result
  If (UBound(arrContent) - LBound(arrContent) + 1) <= 0 Then
      MsgBox "no entries because no connections are implemented"
  End If
End Sub
```

See also

GetTag Function (Page 2571)
DeleteTag Function (Page 2570)
CreateTag Function (Page 2568)
CommitTag Function (Page 2566)
CloseTag Function (Page 2565)
VBA in Tag Management (Page 2562)

3.6.2.3 VBA im Tag Logging

VBA in Tag Logging

Introduction

VBA allows you to create process value archives, archive tags and triggers directly from the program code, modify them, and delete them.

Note

You should not have or should not open the "Tag Logging" editor when editing with VBA.
Principle

When you have created the instance of the "HMIGO" class, the following functions are available to you to access Tag Logging:

- CloseTlgArchive
- CloseTlgTag
- CloseTlgTrigger
- CommitTlgArchive
- CommitTlgTag
- CommitTlgTrigger
- CreateTlgArchive
- CreateTlgTag
- CreateTlgTrigger
- DeleteTlgArchive
- DeleteTlgTag
- DeleteTlgTrigger
- GetTlgArchive
- GetTlgTag
- GetTlgTrigger
- ListTlgArchive
- ListTlgTag
- ListTlgTrigger

The following enumerations are available for the parameter supply of these functions:

- HMIGO_TLG_ARCHIVE_TYPE
- HMIGO_TLG_ARCHIVE_LIST_TYPE
- HMIGO_TLG_TAG_TYPE
- HMIGO_TLG_TAG_LIST_TYPE
- HMIGO_TLG_TRIGGER_BASE
- HMIGO_TLG_TRIGGER_LIST_TYPE
Direct Access to the Object Properties

You can also access the parameters of the above-mentioned functions directly in VBA by means of the following object properties. The column "is used in" will display whether you will be able to access the object property in the process value archive (P) and/or in the compressed archive (V).

<table>
<thead>
<tr>
<th>Object property</th>
<th>Description</th>
<th>Read/Write</th>
<th>is used in</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectStateTlgArchive</td>
<td>Returns the object state for the archive via the enumeration &quot;HMIGO_OBJECT_STATE&quot;. Further information on this enumeration can be found in this documentation under &quot;VBA in other WinCC Editors&quot;.</td>
<td>Yes/no</td>
<td>P, V</td>
</tr>
<tr>
<td>ObjectStateTlgTag</td>
<td>Returns the object state for the archive tag via the enumeration &quot;HMIGO_OBJECT_STATE&quot;.</td>
<td>Yes/no</td>
<td>P, V</td>
</tr>
<tr>
<td>TlgArchiveAccessLevelRead</td>
<td>The authorization level for reading.</td>
<td>Yes/no</td>
<td>P, V</td>
</tr>
<tr>
<td>TlgArchiveAccessLevelWrite</td>
<td>The authorization level for writing.</td>
<td>Yes/no</td>
<td>P, V</td>
</tr>
</tbody>
</table>
| TlgArchiveArchiveState   | Specifies whether archiving is disabled or enabled at system startup. Possible values of the enum "HMIGO_TLG_ARCHIVE_STATE":  
  ● TLG_ARCHIVE_STATE_LOCKED (1)  
  ● TLG_ARCHIVE_STATE_ACTIVATED (0) | Yes/yes    | P, V       |
| TlgArchiveBufferSize     | Specifies the number of records for a short-term archive.                   | Yes/yes    | P          |
| TlgArchiveBufferType     | Specifies the tag storage location. The possible types of the enum "HMIGO_TLG_ARCHIVE_BUFFER_TYPE":  
  ● TLG_ARCHIVE_BUFFER_TYPE_DISK (2)  
  ● TLG_ARCHIVE_BUFFER_TYPE_RAM (1) | Yes/yes    | P          |
| TlgArchiveCompressRange  | Specifies the compression time period. Name of the timer, greater than or equal to 1, defined under "Times" in the Tag Logging editor. Since the format is a string, it is language dependent.  
  Can be determined via the function "ListTlgArchive(TLG_ARCHIVE_TRIGGER_NAMES, arrTrigger)" | Yes/yes    | V          |
| TlgArchiveCompressType   | Specifies the algorithm for compressing the values. The possible types of the enum "HMIGO_TLG_ARCHIVE_COMPRESS_TYPE":  
  ● TLG_COMPRESS_TYPE_CALC (1)  
  ● TLG_COMPRESS_TYPE_CALC_COPY (2)  
  ● TLG_COMPRESS_TYPE_CALC_DEL (3)  
  ● TLG_COMPRESS_TYPE_CALC_COPY_DEL (4) | Yes/yes    | V          |
<p>| TlgArchiveFlags          | Used internally.                                                            |            |            |
| TlgArchiveName           | Name of the process value archive or compressed archive.                   | Yes/no     | P, V       |</p>
<table>
<thead>
<tr>
<th>Object property</th>
<th>Description</th>
<th>Read/Write</th>
<th>is used in</th>
</tr>
</thead>
</table>
| TlgArchiveQCRActive     | Specifies for the compressed archive whether weighted quality codes are used during archiving. The possible types of the enum "HMIGO_TLG_QCR_ACTIVE_FLAGS":  
  • TLG_QCR_ALL (15)  
  • TLG_QCR_BAD (1)  
  • TLG_QCR_GOOD_CASCADED (8)  
  • TLG_QCR_GOOD_NONCASCadedD (4)  
  • TLG_QCR_OFF (0)  
  • TLG_QCR_UNCERTAIN (2) |            | V          |
| TlgArchiveQCRBad        | If weighted quality codes are used and the "Bad" option is activated, you define for the compression archive the percentage as of which the "Bad" state of process values is archived in the compression tag. |            | V          |
| TlgArchiveQCRGoodCascade | If weighted quality codes are used and the "Good (Cascade)" option is activated, you define for the compression tag the percentage as of which the "Good (Cascade)" state of process values is archived in the compression tag. |            | V          |
| TlgArchiveQCRGoodNonCascade | If weighted quality codes are used and the "Good (Non-Cascade)" option is activated, you define for the compression archive the percentage as of which the "Good (Non-Cascade)" state of process values is archived in the compression tag. |            | V          |
| TlgArchiveQCRUncertain  | If weighted quality codes are used and the "Uncertain" option is activated, you define for the compression archive the percentage as of which the "Uncertain" state of process values is archived in the compression tag. |            | V          |
| TlgArchiveType          | Specifies whether the archive is a process value archive or a compressed archive.                                                                                                                             | Yes/no     | P, V       |
| TlgTagAliasName         | The alternative name by means of which the tag can be addressed (alias).                                                                                                                                       | Yes/yes    | P          |
| TlgTagArchiveName       | Name of the archive.                                                                                                                                                                                          | Yes/no     | P, V       |
| TlgTagArchiving         | Specifies the acquisition type. Possible values of the enum "HMIGO_TLG_TAG_ARCHIVING":  
  • TLG_TAGACYCLIC (8388609)  
  • TLG_TAG_CYCLIC_CONTINUOUS (8388610)  
  • TLG_TAG_CYCLIC_SELECTIVE (8388612)  
  • TLG_TAG_ON_EVERY_CHANGE (8388616) | Yes/yes    | P          |
| TlgTagArchivingState    | Specifies whether archiving is enabled or disabled at system startup. Possible values of the enum "HMIGO_TLG_TAG_ARCHIVING_STATE":  
  • TLG_TAG_LOCKED (1)  
  • TLG_TAG_ACTIVATED (0) | Yes/yes    | P, V       |
<p>| TlgTagConvertModule     | Name of the conversion DLL used for data conversion.                                                                                                                                                         | Yes/yes    | P          |</p>
<table>
<thead>
<tr>
<th>Object property</th>
<th>Description</th>
<th>Read/Write</th>
<th>is used in</th>
</tr>
</thead>
<tbody>
<tr>
<td>TlgTagFlags</td>
<td>Possible values of the enum &quot;HMI-GO_TLG_TAG_FLAGS&quot;:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_LONGTERM_DISABLED (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_NOFLAGS (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTagHysterese</td>
<td>Value for the hysteresis by means of which a check is carried out to establish whether a value has changed.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagLowerLimit</td>
<td>Value for the scaling of the tag's lower limit.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagMethodType</td>
<td>Specifies the method by which the value is edited before archiving. Possible values of the enum &quot;HMI-GO_TLG_TAG_METHOD_TYPE&quot;:</td>
<td>Yes/yes</td>
<td>P, V</td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_ACTUAL (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_SUM (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_MaxValue (5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_MinValue (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_AVERAGE (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTagName</td>
<td>Name of the archive tag.</td>
<td>Yes/no</td>
<td>P, V</td>
</tr>
<tr>
<td>TlgTagNameCompressArchive</td>
<td>In the case of compressed archives, contains the name of the source archive.</td>
<td>Yes/yes</td>
<td>V</td>
</tr>
<tr>
<td>TlgTagNameCompressTag</td>
<td>In the case of compressed archives, contains the name of the source tag.</td>
<td>Yes/yes</td>
<td>V</td>
</tr>
<tr>
<td>TlgTagNameProcTag</td>
<td>Name of the process tag from which the value to be acquired is taken.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagNameRawValue</td>
<td>In the cast of process-controlled archives, contains the name of the raw-data tag.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagOnChange</td>
<td>Specifies whether archiving is to be carried out in the event of a change. Possible values of the enum &quot;HMI-GO_TLG_TAG_ON_CHANGE&quot;:</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_EVERY_VALUE (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_RELATIVE_HYSTERESE (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_ABSOLUTE_HYSTERESE (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTagOnError</td>
<td>Specifies whether, in the event of a problem, the most recently acquired value or the substitute value is saved. Possible values of the enum &quot;HMI-GO_TLG_TAG_ON_ERROR&quot;:</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_LAST_VALUE (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● TLG_TAG_SUBSTITUTE (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTagPreviousOSGUID</td>
<td>Used internally.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object property</td>
<td>Description</td>
<td>Read/Write</td>
<td>is used in</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>TlgTagQCRActive</td>
<td>Specifies for the compression tag whether weighted quality codes are used during archiving. The possible types of the enum &quot;HMIGO_TLG_QCR_ACTIVE_FLAGS&quot;: &lt;ul&gt;&lt;li&gt;TLG_QCR_ALL (15)&lt;/li&gt;&lt;li&gt;TLG_QCR_BAD (1)&lt;/li&gt;&lt;li&gt;TLG_QCR_GOOD_CASCaded (8)&lt;/li&gt;&lt;li&gt;TLG_QCR_GOOD_NONCASCaded (4)&lt;/li&gt;&lt;li&gt;TLG_QCR_OFF (0)&lt;/li&gt;&lt;li&gt;TLG_QCR_UNCERTAIN (2)&lt;/li&gt;&lt;/ul&gt;</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>TlgTagQCRBad</td>
<td>If weighted quality codes are used and the &quot;Bad&quot; option is activated, you define for the compression tag the percentage as of which the &quot;Bad&quot; state of process values is archived in the compression tag.</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>TlgTagQCRGoodCascade</td>
<td>If weighted quality codes are used and the &quot;Good (Cascade)&quot; option is activated, you define for the compression tag the percentage as of which the &quot;Good (Cascade)&quot; state of process values is archived in the compression tag.</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>TlgTagQCRGoodNonCascade</td>
<td>If weighted quality codes are used and the &quot;Good (Non-Cascade)&quot; option is activated, you define for the compression tag the percentage as of which the &quot;Good (Non-Cascade)&quot; state of process values is archived in the compression tag.</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>TlgTagQCRUncertain</td>
<td>If weighted quality codes are used and the &quot;Uncertain&quot; option is activated, you define for the compression tag the percentage as of which the &quot;Uncertain&quot; state of process values is archived in the compression tag.</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDCompressDeviation</td>
<td>Specifies the absolute or relative value of the deviation, which is permitted for the calculation of the increase by the algorithm. Basic value is the process value saved last.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDCompression</td>
<td>Specifies whether the swinging door compression is activated.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDLowLimit</td>
<td>Specifies the low limit of the swinging door compression distribution when a relative deviation is activated.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDRelativeDecision</td>
<td>Specifies whether the relative value of the deviation is taken into consideration at the swinging door algorithm.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDtMax</td>
<td>Specifies the maximum duration between two archived values as the detection limit for swinging door compression.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDtMin</td>
<td>Specifies the minimum duration between two archived values as the detection limit for swinging door compression.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagSDUpperLimit</td>
<td>Specifies the upper limit of the swinging door compression distribution when a relative deviation is activated.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTagStartEvent</td>
<td>Name of the tag by means of which the start of archiving is checked.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>Object property</td>
<td>Description</td>
<td>Read/Write</td>
<td>is used in</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>TlgTagStartTriggerFunction</td>
<td>Specifies the name of a script function by means of which a check is carried out for a start event for the start of archiving.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagStopEvent</td>
<td>Name of the tag by means of which the stopping of archiving is checked.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagStopTriggerFunction</td>
<td>Specifies the name of a script function by means of which a check is carried out for a stop event for the start of archiving.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagTriggerArchiving</td>
<td>Name of the timer for the archiving cycle.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagTriggerFactor</td>
<td>Contains the factor for the display cycle as a multiple of the archiving cycle.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagTriggerFunction</td>
<td>Specifies the name of a script function for the dynamic switching of the acquisition and archiving cycles.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagTriggerScan</td>
<td>Name of the timer for the acquisition cycle</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagTriggerType</td>
<td>Specifies how archiving is carried out at a signal change. Possible values of the enum &quot;HMIGO_TLG_TAG_TRIGGER_TYPE&quot;:&lt;br&gt;● TLG_TAG_FROM_0_TO_1 (2)&lt;br&gt;● TLG_TAG_FROM_1_TO_0 (3)&lt;br&gt;● TLG_TAG_ALWAYS (4)&lt;br&gt;● TLG_TAG_EVERY_CHANGE (1)</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTagType</td>
<td>Specifies the tag type. The possible types of the enum &quot;HMIGO_TLG_TAG_TYPE&quot;:&lt;br&gt;● TLG_TAG_TYP_ANALOG (65537)&lt;br&gt;● TLG_TAG_TYP_BINARY (65538)&lt;br&gt;● TLG_TAG_TYP_PROCESS (65544)&lt;br&gt;● TLG_TAG_TYP_COMPRESS (65540)</td>
<td>Yes/yes</td>
<td>P, V</td>
</tr>
<tr>
<td>TlgTagUpperLimit</td>
<td>Value for the scaling of the tag's upper limit.</td>
<td>Yes/yes</td>
<td>P</td>
</tr>
<tr>
<td>TlgTriggerBase</td>
<td>Time base for the trigger. Possible values of the enum &quot;HMIGO_TLG_TRIGGER_BASE&quot;:&lt;br&gt;● TLG_TRIGGER_BASE_250MS (250)&lt;br&gt;● TLG_TRIGGER_BASE_500MS (500)&lt;br&gt;● TLG_TRIGGER_BASE_DAY (&amp;H5265C00)&lt;br&gt;● TLG_TRIGGER_BASE_HOUR (&amp;H36EE80)&lt;br&gt;● TLG_TRIGGER_BASE_MIN (&amp;HEA60)&lt;br&gt;● TLG_TRIGGER_BASE_SEC (&amp;H3E8)</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>TlgTriggerCreatorID</td>
<td>Used internally.</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>TlgTriggerFactor</td>
<td>Integer factor that is taken into consideration for the trigger together with the time base.</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>TlgTriggerName</td>
<td>Name of the trigger.</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>TlgTriggerScheduleDayOfMonth</td>
<td>Specifies the day for the trigger for the &quot;Monthly&quot; time series.</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>Object property</td>
<td>Description</td>
<td>Read/Write</td>
<td>is used in</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>TlgTriggerScheduleDaysOfWeek</td>
<td>Specifies the days for the trigger for the &quot;Weekly&quot; time series. Possible values of the enum &quot;HMIGO_TLG_TRIGGER_SCHEDULE_DAYSOFWEEK&quot;:</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_EVERY_DAY (127)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_FRI DAY (32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_MONDAY (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_NO_DAY (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_SATURDAY (64)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_SUNDAY (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_THURSDAY (16)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_TUESDAY (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_WEDNESDAY (8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTriggerScheduleInterval</td>
<td>Specifies the interval for the trigger for the calendar times.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerScheduleMonthOfYear</td>
<td>Specifies the month for the trigger for the &quot;Yearly&quot; time series.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerScheduleType</td>
<td>Defines the type of trigger. Possible time series of the enum &quot;HMIGO_TLG_TRIGGER_SCHEDULE_TYPE&quot;:</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_TYPE_CYCLIC (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_TYPE_DAILY (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_TYPE_MONTHLY (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_TYPE_WEEKLY (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TLG_TRIGGER_SCHEDULE_TYPE_YEARLY (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartByShutdown</td>
<td>The trigger is initiated additionally when the system is shut down regardless of the configured triggers.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartByStartup</td>
<td>The trigger is initiated additionally when the system is started up regardless of the configured triggers.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartDay</td>
<td>Specifies the day for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartHour</td>
<td>Specifies the hour for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartMilliSecond</td>
<td>Specifies the milliseconds for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartMinute</td>
<td>Specifies the minute for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartMonth</td>
<td>Specifies the month for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartSecond</td>
<td>Specifies the seconds for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>TlgTriggerStartYear</td>
<td>Specifies the year for the starting point of the trigger.</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>
See also

ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
DeleteTlgArchive Function (Page 2598)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Other WinCC Editors (Page 2560)
CloseTlgTrigger function (Page 2584)
CommitTlgTrigger function (Page 2589)
CreateTlgTrigger function (Page 2596)
DeleteTlgTrigger function (Page 2600)
GetTlgTrigger function (Page 2604)
ListTlgTrigger function (Page 2607)

CloseTlgArchive Function

Description

Closes the process value or compressed archive which is open.

Note

Modified parameters are not saved.

Syntax

Expression.CloseTlgArchive()

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

--
Example:

Sub CloseTlgArchive()
  ' HMIGO_006
  ' procedure to close an archive
  ' the archive need to be created before
  ' declarations
  Dim objHMIGO As HMIGO
  Dim strArchiveName As String
  Set objHMIGO = New HMIGO
  strArchiveName = "NewArchive"
  ' current status is "EMPTY"
  MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
  ' open archive
  objHMIGO.GetTlgArchive strArchiveName
  ' current status is "OPENED"
  MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
  ' close archive
  objHMIGO.CloseTlgArchive
  ' current status is "EMPTY"
  MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"

  Set objHMIGO = Nothing
End Sub

See also

ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
DeleteTlgArchive Function (Page 2598)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
VBA in Tag Logging (Page 2573)
CloseTlgTag Function

Description
Closes the archive tag which is open.

Note
Modified parameters are not saved.

Syntax
Expression.CloseTlgTag()

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters
-

Example:

Sub CloseTlgTag()
' HMIGO_007
' procedure to close a tag logging tag
' the archive need to be created before
' the tag logging tag need to be created before
' declarations
Dim objHMIGO As HMIGO
Dim strArchiveName As String
Dim strTlgTagName As String
Set objHMIGO = New HMIGO
strArchiveName = "NewArchive"
strTlgTagName = "NewTag"

'current status is "EMPTY"
MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Tag"
'open/ get tag logging tag
objHMIGO.GetTlgTag strArchiveName, strTlgTagName
'current status is "OPENED"
MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
'close tag logging tag
objHMIGO.CloseTlgTag
'current status is "EMPTY"
MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
Set objHMIGO = Nothing
End Sub
CloseTlgTrigger function

Description
Closes the opened trigger.

Note
Modified parameters are not saved.

Syntax
Expression.CloseTlgTrigger()

Expression
Required. An expression which returns a "HMIGO" type object.

Parameter
--
CommitTlgArchive Function

Description

Writes the changed parameters of the specified archive to WinCC.

Note

If further parameters are changed after a CommitTlgArchive call, write the changes to WinCC by calling this function again.

Syntax

Expression.CommitTlgArchive()

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

--
Example:

Sub CommitTlgArchive()
    ' HMIGO_008
    ' procedure to change a property of an archive
    ' the archive need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    'open archive
    objHMIGO.GetTlgArchive strArchiveName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    'change a property
    objHMIGO.TlgArchiveArchiveState = TLG_STATE_LOCKED
    'current status is "MODIFIED"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    'commit archive
    objHMIGO.CommitTlgArchive
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"

    Set objHMIGO = Nothing
End Sub

See also

DeleteTlgArchive Function (Page 2598)
ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Tag Logging (Page 2573)
CommitTlgTag Function

Description
Writes the changed parameters of the specified archive tag to WinCC.

Note
If further parameters are changed after a CommitTlgTag call, write the changes to WinCC by calling this function again.

Syntax
Expression.CommitTlgTag()

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters
--
Example:

Sub CommitTlgTag()
    ' HMIGO_009
    ' procedure to change a property of a tag logging tag
    ' the archive need to be created before
    ' the tag logging tag need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Dim strTlgTagName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    strTlgTagName = "NewTag"

    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Tag"
    'open/ get tag logging tag
    objHMIGO.GetTlgTag strArchiveName, strTlgTagName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
    'change a property
    objHMIGO.TlgTagArchiving = TLG_TAG_ON_EVERY_CHANGE
    'current status is "MODIFIED"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
    'commit tag logging tag
    objHMIGO.CommitTlgTag
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
    Set objHMIGO = Nothing
End Sub

See also

ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
DeleteTlgArchive Function (Page 2598)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Tag Logging (Page 2573)
CommitTlgTrigger function

Description

Writes the changed parameters of the specified trigger to WinCC.

Note

If further parameters are changed after a CommitTlgTrigger call, write the changes to WinCC by calling this function again.

Syntax

Expression.CommitTlgTrigger()

Expression

Required. An expression which returns a "HMIGO" type object.

Parameter

--

CreateTlgArchive Function

Description

Creates a process value archive or compressed archive.

Syntax

Expression.CreateTlgArchive(ArchiveName,ArchiveType)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive to be created</td>
</tr>
<tr>
<td>ArchiveType (HMIGO_TLG_ARCHIVE_TYPE)</td>
<td>Type of the archive. The possible types are:</td>
</tr>
<tr>
<td></td>
<td>• TLG_PROCESSARCHIVE (131073) for a process value archive</td>
</tr>
<tr>
<td></td>
<td>• TLG_COMPRESSARCHIVE (131074) for a compressed archive</td>
</tr>
</tbody>
</table>
Default Values when Creating a New Tag Archive

The following table indicates the default values that are entered when a new process value archive or compressed archive is created. These values can be modified later and written using the CommitTlgArchive function.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value (Enum Name =&gt; Value)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TlgArchiveAccessLevel-Read</td>
<td>0</td>
<td>Without authorization level</td>
</tr>
<tr>
<td>TlgArchiveAccessLevel-Write</td>
<td>0</td>
<td>Without authorization level</td>
</tr>
<tr>
<td>TlgArchiveArchiveState</td>
<td>TLG_ARCHIVE_STATE_ACTIVATED (0)</td>
<td>Archiving is started at start of Runtime.</td>
</tr>
<tr>
<td>TlgArchiveBufferSize</td>
<td>1000</td>
<td>Number of data records</td>
</tr>
<tr>
<td>TlgArchiveBufferType</td>
<td>TLG_ARCHIVE_BUFFER_TYPE_DISK (2)</td>
<td>The values are stored on hard disk in the database.</td>
</tr>
<tr>
<td>TlgArchiveCompress-Range</td>
<td>&quot;1 Tag&quot;. This string must be created individually for each language (e.g. English: &quot;1 day&quot;)</td>
<td>Corresponds to exactly one day. Only relevant in the case of compressed tags. Special Feature: the user is responsible for values &gt;= 1 minute</td>
</tr>
<tr>
<td>TlgArchiveCompressType</td>
<td>TLG_COMPRESS_TYPE_CALC (1)</td>
<td>Only calculate compression values. Only relevant in the case of compressed tags.</td>
</tr>
</tbody>
</table>

Enum HMIGO_TLG_ARCHIVE_STATE

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_ARCHIVE_STATE_LOCKED (1)</td>
<td>Archiving is disabled at system startup.</td>
</tr>
<tr>
<td>TLG_ARCHIVE_STATE_ACTIVATED (0)</td>
<td>Archiving is started at start of Runtime.</td>
</tr>
</tbody>
</table>

Enum HMIGO_TLG_ARCHIVE_BUFFER_TYPE

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_ARCHIVE_BUFFER_TYPE_DISK (2)</td>
<td>The values are archived on hard disk.</td>
</tr>
<tr>
<td>TLG_ARCHIVE_BUFFER_TYPE_RAM (1)</td>
<td>The values are only archived in working memory.</td>
</tr>
</tbody>
</table>

Enum HMIGO_TLG_ARCHIVE_COMPRESS_TYPE

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_COMPRESS_TYPE_CALC (1)</td>
<td>Only the compression values are calculated.</td>
</tr>
<tr>
<td>TLG_COMPRESS_TYPE_CALC_COPY (2)</td>
<td>The compression values are calculated and the original values copied.</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_COMPRESS_TYPE_CALC_DEL (3)</td>
<td>The compression values are calculated and the original values then deleted.</td>
</tr>
<tr>
<td>TLG_COMPRESS_TYPE_CALC_COPY_DEL (4)</td>
<td>The compression values are calculated and the original values copied and then deleted.</td>
</tr>
</tbody>
</table>

### Example:

```vba
Sub CreateTlgArchive()
    ' HMIGO_010
    ' procedure to create an archive
    ' the archive must not be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    'create tag logging archive
    objHMIGO.CreateTlgArchive strArchiveName, TLG_PROCESSARCHIVE
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    Set objHMIGO = Nothing
End Sub
```

### See also

- GetTlgArchive Function (Page 2601)
- ListTlgTag Function (Page 2606)
- ListTlgArchive Function (Page 2604)
- DeleteTlgTag Function (Page 2599)
- DeleteTlgArchive Function (Page 2598)
- CreateTlgTag Function (Page 2592)
- CommitTlgTag Function (Page 2587)
- CommitTlgArchive Function (Page 2585)
- CloseTlgTag Function (Page 2583)
- CloseTlgArchive Function (Page 2581)
- VBA in Tag Logging (Page 2573)
CreateTlgTag Function

Description

Creates a new archive tag.

Syntax

Expression.CreateTlgTag(ArchiveName, TagName, [TagType])

Expression

Required. An expression which returns a "HMIGO" type object.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of an existing archive in which the tag is to be entered</td>
</tr>
<tr>
<td>TagName (string)</td>
<td>Name of the tag to be created</td>
</tr>
</tbody>
</table>
| TagType (HMIGO_TLG_TAG_TYPE, optional) | Specifies the type of the tag. If no type is specified, the default value TLG_VAR_TYP_ANALOG is entered. Possible types are:  
  - TLG_VAR_TYP_ANALOG (65537) for an analog tag.  
  - TLG_VAR_TYP_BINARY (65538) for a binary tag.  
  - TLG_VAR_TYP_PROCESS (65544) for a process tag.  
  - TLG_VAR_TYP_COMPRESS (65540) for a compression tag. |

Default Values When a New Archive Tag Is Created

The following table indicates the default values that are entered when a new archive tag is created. These values can be modified later and written using the CommitTlgTag function.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default Value (Enum Name =&gt; Value)</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TlgTagType</td>
<td>TLG_VAR_TYP_ANALOG (65537)</td>
<td>Acquired by means of an analog data manager tag</td>
</tr>
<tr>
<td>TlgTagArchiving</td>
<td>TLG_TAG_CYCLIC_CONTINUOUS (8388610)</td>
<td>Cyclic, continuous acquisition</td>
</tr>
<tr>
<td>TlgTagArchivingState</td>
<td>TLG_TAG_ACTIVATED (0)</td>
<td>Archiving is started at start of Runtime.</td>
</tr>
<tr>
<td>TlgTagTriggerScan</td>
<td>1 second</td>
<td>Please note that &quot;1 second&quot; is only the name of the trigger. You must ensure yourself that the trigger exists and actually has a cycle of 1 s.</td>
</tr>
<tr>
<td>Property</td>
<td>Default Value (Enum Name =&gt; Value)</td>
<td>Note</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>TlgTagTriggerArchiving</td>
<td>1 second</td>
<td>Please note that &quot;1 second&quot; is only the name of the trigger. You must ensure yourself that the trigger exists and actually has a cycle of 1 s.</td>
</tr>
<tr>
<td>TlgTagTriggerFactor</td>
<td>1</td>
<td>The display cycle and archiving cycle are identical.</td>
</tr>
<tr>
<td>TlgTagOnError</td>
<td>TLG_TAG_LAST_VALUE (1)</td>
<td>The last valid value is taken as the substitute value.</td>
</tr>
<tr>
<td>TlgTagTriggerType</td>
<td>TLG_TAG_ALWAYS (4)</td>
<td>Every value is archived.</td>
</tr>
<tr>
<td>TlgTagMethodType</td>
<td>TLG_TAG_ACTUAL (1)</td>
<td>No editing. The value is accepted immediately.</td>
</tr>
<tr>
<td>TlgTagStartTriggerFunction</td>
<td>No function specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagStopTriggerFunction</td>
<td>No function specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagTriggerFunction</td>
<td>No function specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagUpperLimit</td>
<td>No value specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagLowerLimit</td>
<td>No value specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagNameCompressArchive</td>
<td>No archive name specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagNameCompressTag</td>
<td>No tag name specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagNameRawValue</td>
<td>No raw data tag specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagNameProcTag</td>
<td>Corresponds to &quot;TagName&quot;</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagOnChange</td>
<td>TLG_TAG_EVERY_VALUE (0)</td>
<td>Every value is archived.</td>
</tr>
<tr>
<td>TlgTagHysterese</td>
<td>0</td>
<td>No check is carried out by means of hysteresis.</td>
</tr>
<tr>
<td>TlgTagAliasName</td>
<td>No value specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagStartEvent</td>
<td>No tag specified</td>
<td>--</td>
</tr>
<tr>
<td>TlgTagStopEvent</td>
<td>No tag specified</td>
<td>--</td>
</tr>
</tbody>
</table>

**List of the enumerators for Tag Logging**

<table>
<thead>
<tr>
<th>Enum types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_TYPE</td>
<td>The passed parameter specifies the type of the tag. The possible types are in the table Enum HMIGO_TLG_TAG_TYPE.</td>
</tr>
<tr>
<td>TLG_TAG_ARCHIVING</td>
<td>The passed parameter specifies the acquisition type. The possible values are in the table Enum HMIGO_TLG_TAG_ARCHIVING.</td>
</tr>
<tr>
<td>TLG_TAG_ARCHIVING_STATE</td>
<td>The passed parameter specifies whether archiving is disabled or enabled at system startup. The possible values are in the table Enum HMIGO_TLG_TAG_ARCHIVING_STATE.</td>
</tr>
<tr>
<td>TLG_TAG_ON_ERROR</td>
<td>The passed parameter specifies which value is stored in the event of a problem: the most recently acquired value or the substitute value. The possible values are in the table Enum HMIGO_TLG_TAG_ON_ERROR.</td>
</tr>
</tbody>
</table>
## TLG_TAG_TRIGGGER_TYPE

The passed parameter specifies how archiving is carried out at a signal change. The possible values are in the table `Enum HMIGO_TLG_TAG_TRIGGGER_TYPE`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_TYP_ANALOG (65537)</td>
<td>Analog tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_BINARY (65538)</td>
<td>Binary tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_PROCESS (65544)</td>
<td>Process tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_COMPRESS (65540)</td>
<td>Compressed archive tag</td>
</tr>
</tbody>
</table>

## TLG_TAG_METHOD_TYPE

The passed parameter specifies the method by which the value is edited before archiving. The possible values are in the table `Enum HMIGO_TLG_TAG_METHOD_TYPE`.

## TLG_TAG_ON_CHANGE

The passed parameter specifies whether archiving is to be carried out in the event of a change. The possible values are in the table `Enum HMIGO_TLG_TAG_ON_CHANGE`.

## Enum HMIGO_TLG_TAG_TYPE

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_TYP_ANALOG (65537)</td>
<td>Analog tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_BINARY (65538)</td>
<td>Binary tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_PROCESS (65544)</td>
<td>Process tag</td>
</tr>
<tr>
<td>TLG_TAG_TYP_COMPRESS (65540)</td>
<td>Compressed archive tag</td>
</tr>
</tbody>
</table>

## Enum HMIGO_TLG_TAG_ARCHIVING

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_ACYCLIC (8388609)</td>
<td>Acyclic acquisition</td>
</tr>
<tr>
<td>TLG_TAG_CYCLIC_CONTINUOUS (8388610)</td>
<td>Cyclic-continuous acquisition</td>
</tr>
<tr>
<td>TLG_TAG_CYCLIC_SELECTIVE (8388612)</td>
<td>Cyclic-selective acquisition</td>
</tr>
<tr>
<td>TLG_TAG_ON_EVERY_CHANGE (8388616)</td>
<td>Acquisition only in the event of a change</td>
</tr>
</tbody>
</table>

## Enum HMIGO_TLG_TAG_ARCHIVING_STATE

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_LOCKED (1)</td>
<td>Acquisition disabled at system startup</td>
</tr>
<tr>
<td>TLG_TAG_ACTIVATED (0)</td>
<td>Acquisition enabled at system startup</td>
</tr>
</tbody>
</table>

## Enum HMIGO_TLG_TAG_ON_ERROR

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_LAST_VALUE (1)</td>
<td>The most recently acquired value is used.</td>
</tr>
<tr>
<td>TLG_TAG_SUBSTITUTE (2)</td>
<td>A substitute value is entered.</td>
</tr>
</tbody>
</table>
## Enum HMIGO_TLG_TAG_TRIGGER_TYPE

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_FROM_0_TO_1 (2)</td>
<td>Signal change from the value 0 to 1.</td>
</tr>
<tr>
<td>TLG_TAG_FROM_1_TO_0 (3)</td>
<td>Signal change from the value 1 to 0.</td>
</tr>
<tr>
<td>TLG_TAG_ALWAYS (4)</td>
<td>Always archive.</td>
</tr>
<tr>
<td>TLG_TAG_EVERY_CHANGE (1)</td>
<td>Archive at every signal change.</td>
</tr>
</tbody>
</table>

## Enum HMIGO_TLG_TAG_METHOD_TYPE

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_ACTUAL (1)</td>
<td>The current value is accepted.</td>
</tr>
<tr>
<td>TLG_TAG_SUM (3)</td>
<td>The sum is formed.</td>
</tr>
<tr>
<td>TLG_TAG_MaxValue (5)</td>
<td>The greatest value is saved.</td>
</tr>
<tr>
<td>TLG_TAG_MinValue (4)</td>
<td>The smallest value is saved.</td>
</tr>
<tr>
<td>TLG_TAG_AVERAGE (2)</td>
<td>The average value is saved.</td>
</tr>
</tbody>
</table>

## Enum HMIGO_TLG_TAG_ON_CHANGE

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TAG_EVERY_VALUE (0)</td>
<td>The current value is accepted.</td>
</tr>
<tr>
<td>TLG_TAG_RELATIVE_HYSTERESE (1)</td>
<td>A hysteresis specified as a percentage is used for the calculation as to whether the value is to be archived.</td>
</tr>
<tr>
<td>TLG_TAG_ABSOLUTE_HYSTERESE (2)</td>
<td>A hysteresis specified as an absolute value is used for the calculation as to whether the value is to be archived.</td>
</tr>
</tbody>
</table>
Sub CreateTlgTag()
    ' HMIGO_011
    ' procedure to create a tag logging tag
    ' the archive need to be created before
    ' the tag logging tag must not be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Dim strTlgTagName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    strTlgTagName = "NewTag"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Tag"
    'create tag logging tag
    objHMIGO.CreateTlgTag strArchiveName, strTlgTagName, TLG_TAG_TYPE_ANALOG
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
    Set objHMIGO = Nothing
End Sub

See also
ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
DeleteTlgArchive Function (Page 2598)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CloseTlgArchive Function (Page 2581)
CloseTlgTag Function (Page 2583)
VBA in Tag Logging (Page 2573)

CreateTlgTrigger function

Description

Creates a new trigger that is used as a timer for the acquisition and archiving cycle.
Syntax

Expression.CreateTlgTrigger(TriggerName, TriggerBase, TriggerFactor)

Expression

Required. An expression which returns a "HMIGO" type object.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerName (String)</td>
<td>Name of the trigger to be created</td>
</tr>
</tbody>
</table>
| TriggerBase (HMIGO_TLG_TRIGGER_BASE) | Time base of the trigger. Possible values:  
  - TLG_TRIGGER_BASE_250MS (250)  
  - TLG_TRIGGER_BASE_500MS (500)  
  - TLG_TRIGGER_BASE_DAY (&H5265C00)  
  - TLG_TRIGGER_BASE_HOUR (&H36EE80)  
  - TLG_TRIGGER_BASE_MIN (&HEA60)  
  - TLG_TRIGGER_BASE_SEC (&H3E8) |
| TriggerFactor | Integer factor that is taken into consideration for the trigger together with the time base. |

Enum HMIGO_TLG_TRIGGER_BASE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TRIGGER_BASE_250MS (250)</td>
<td>The time base is &quot;250 ms&quot;.</td>
</tr>
<tr>
<td>TLG_TRIGGER_BASE_500MS (500)</td>
<td>The time base is &quot;500 ms&quot;.</td>
</tr>
<tr>
<td>TLG_TRIGGER_BASE_DAY (&amp;H5265C00)</td>
<td>The time base is &quot;1 day&quot;.</td>
</tr>
<tr>
<td>TLG_TRIGGER_BASE_HOUR (&amp;H36EE80)</td>
<td>The time base is &quot;1 hour&quot;.</td>
</tr>
<tr>
<td>TLG_TRIGGER_BASE_MIN (&amp;HEA60)</td>
<td>The time base is &quot;1 minute&quot;.</td>
</tr>
<tr>
<td>TLG_TRIGGER_BASE_SEC (&amp;H3E8)</td>
<td>The time base is &quot;1 second&quot;.</td>
</tr>
</tbody>
</table>

Enum HMIGO_TLG_TRIGGER_SCHEDULE_DAYSOFWEEK

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_EVERY_DAY (127)</td>
<td>Every day is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_FRIDAY (32)</td>
<td>Friday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_MONDAY (2)</td>
<td>Monday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_NO_DAY (0)</td>
<td>No day is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_SATURDAY (64)</td>
<td>Saturday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_SUNDAY (1)</td>
<td>Sunday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_THURSDAY (16)</td>
<td>Thursday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_TUESDAY (4)</td>
<td>Tuesday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_DAYSOFWEEK_WEDNESDAY (8)</td>
<td>Wednesday is used for the &quot;Weekly&quot; trigger.</td>
</tr>
</tbody>
</table>
Enum HMIGO_TLG_TRIGGER_SCHEDULE_TYPE

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLG_TRIGGER_SCHEDULE_TYPE_CYCLIC (0)</td>
<td>The time series &quot;Standard (cyclic)&quot; is used for the trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_TYPE_DAILY (1)</td>
<td>The time series &quot;Daily&quot; is used for the trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_TYPE_MONTHLY (3)</td>
<td>The time series &quot;Monthly&quot; is used for the trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_TYPE_WEEKLY (2)</td>
<td>The time series &quot;Weekly&quot; is used for the trigger.</td>
</tr>
<tr>
<td>TLG_TRIGGER_SCHEDULE_TYPE_YEARLY (4)</td>
<td>The time series &quot;Yearly&quot; is used for the trigger.</td>
</tr>
</tbody>
</table>

DeleteTlgArchive Function

Description

Deletes the specified archive.

Syntax

Expression.DeleteTlgArchive(ArchiveName)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive to be deleted. Archive tags contained in the archive are also deleted.</td>
</tr>
</tbody>
</table>

Example:

```vba
Sub DeleteTlgArchive()
    ' HMIGO_012
    ' procedure to delete an archive
    ' the archive need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"

    'delete tag logging archive
    objHMIGO.DeleteTlgArchive strArchiveName
    Set objHMIGO = Nothing
End Sub
```
See also

ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Tag Logging (Page 2573)

DeleteTlgTag Function

Description

Deletes the specified archive tag.

Syntax

Expression.DeleteTlgTag(ArchiveName,TagName)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive containing the archive tag to be deleted</td>
</tr>
<tr>
<td>TagName (string)</td>
<td>Name of the archive tag to be deleted.</td>
</tr>
</tbody>
</table>
Example:

Sub DeleteTlgTag()
    ' HMIGO_013
    ' procedure to delete a tag logging tag
    ' the archive need to be created before
    ' the tag logging tag need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Dim strTlgTagName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    strTlgTagName = "NewTag"

    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Tag"
    'delete tag logging tag
    objHMIGO.DeleteTlgTag strArchiveName, strTlgTagName
    Set objHMIGO = Nothing
End Sub

See also

ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgArchive Function (Page 2598)
CreateTlgTag Function (Page 2592)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Tag Logging (Page 2573)

DeleteTlgTrigger function

Description

Deletes the specified trigger.
Syntax

Expression.DeleteTlgTrigger(TriggerName)

Expression

Required. An expression which returns a "HMIGO" type object.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerName (String)</td>
<td>Name of the trigger that is deleted.</td>
</tr>
</tbody>
</table>

GetTlgArchive Function

Description

Reads in the parameters of the specified archive.

You can change or read the parameters by means of the object properties. You will find a list of the available object properties in this documentation under "VBA in TagLogging".

syntax

Expression.GetTlgArchive(ArchiveName)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive whose values are to be read in.</td>
</tr>
</tbody>
</table>
Example:

Sub GetTlgArchive()
    ' HMIGO_014
    ' procedure to open an archive
    ' the archive need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    ' current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    ' open/ get tag logging archive
    objHMIGO.GetTlgArchive strArchiveName
    ' current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgArchive, vbOKOnly, "Status Tlg Archive"
    Set objHMIGO = Nothing
End Sub

See also

CreateTlgTag Function (Page 2592)
ListTlgTag Function (Page 2606)
ListTlgArchive Function (Page 2604)
GetTlgArchive Function (Page 2601)
DeleteTlgTag Function (Page 2599)
DeleteTlgArchive Function (Page 2598)
CreateTlgArchive Function (Page 2589)
CommitTlgTag Function (Page 2587)
CommitTlgArchive Function (Page 2585)
CloseTlgTag Function (Page 2583)
CloseTlgArchive Function (Page 2581)
VBA in Tag Logging (Page 2573)

GetTlgTag Function

Description

Reads in the parameters of the specified archive tag.
You can change or read the parameters by means of the object properties. You will find a list of the available object properties in this documentation under "VBA in TagLogging".
**syntax**

```
Expression.GetTlgTag(ArchiveName,TagName)
```

**Expression**

Necessary. An expression which returns a "HMIGO" type object.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive containing the archive tag.</td>
</tr>
<tr>
<td>TagName</td>
<td>Name of the archive tag whose parameters are to be read in.</td>
</tr>
</tbody>
</table>

**Example:**

```
Sub GetTlgTag()
    ' HMIGO_015
    ' procedure to close a tag logging tag
    ' the archive need to be created before
    ' the tag logging need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strArchiveName As String
    Dim strTlgTagName As String
    Set objHMIGO = New HMIGO
    strArchiveName = "NewArchive"
    strTlgTagName = "NewTag"

    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Tag"
    'open/ get tag logging tag
    objHMIGO.GetTlgTag strArchiveName, strTlgTagName
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateTlgTag, vbOKOnly, "Status Tlg Archive"
    Set objHMIGO = Nothing
End Sub
```

**See also**

- CreateTlgTag Function (Page 2592)
- ListTlgTag Function (Page 2606)
- ListTlgArchive Function (Page 2604)
- GetTlgArchive Function (Page 2601)
- DeleteTlgTag Function (Page 2599)
- DeleteTlgArchive Function (Page 2598)
GetTlgTrigger function

Description
Reads in the parameters of the specified trigger.
You can change or read the parameters by means of the object properties. You will find a list of the available object properties in this documentation under "VBA in TagLogging".

Syntax
Expression.GetTlgTrigger(TriggerName)

Expression
Required. An expression which returns a "HMIGO" type object.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerName (String)</td>
<td>Name of the trigger whose values are read in.</td>
</tr>
</tbody>
</table>

ListTlgArchive Function

Description
Alternatively, the ListTlgArchive function returns the following Tag Logging values in a list:
- All existing Tag Logging archives
- All existing cycles / timers

Syntax
Expression.ListTlgArchive(ListType,pListArray,[Filter])

Expression
Necessary. An expression which returns a "HMIGO" type object.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListType (HMIGO_TLG_ARCHIVE_LIST_TYPE)</td>
<td>Defines which content should be returned in a list. Possibilities are:</td>
</tr>
<tr>
<td></td>
<td>• TLG_ARCHIVE_NAMES (1) All created Tag Logging archives</td>
</tr>
<tr>
<td></td>
<td>• TLG_ARCHIVE_TRIGGER_NAMES (2) All created cycles / timers</td>
</tr>
<tr>
<td>pListArray (Variant)</td>
<td>List with the requested content.</td>
</tr>
<tr>
<td>Filter (String)</td>
<td>Filters can be set optionally. A trigger name can be used as a filter. Wildcards &quot;*&quot; and &quot;?&quot; are also possible.</td>
</tr>
</tbody>
</table>

Example:

In the following example, a check is made whether archives are configured:

```vba
Sub ReadTlgArchives()
    'HMIGO_028
    'read content in tag logging
    'no archives are implemented
    Dim objHMIGO As New HMIGO
    Dim varRange As Variant
    'read all tlg archives
    objHMIGO.ListTlgArchive TLG_ARCHIVE_NAMES, arrContent
    'check result
    If (UBound(arrContent) - LBound(arrContent) + 1) <= 0 Then
        MsgBox "no entries because no tag logging archives are implemented"
    End If
End Sub
```

See also

- ListTlgTag Function (Page 2606)
- GetTlgArchive Function (Page 2601)
- DeleteTlgTag Function (Page 2599)
- DeleteTlgArchive Function (Page 2598)
- CreateTlgTag Function (Page 2592)
- CreateTlgArchive Function (Page 2589)
- CommitTlgTag Function (Page 2587)
- CommitTlgArchive Function (Page 2585)
- CloseTlgTag Function (Page 2583)
ListTlgTag Function

Description

The ListTlgTag function returns all the tags created in a Tag Logging archive in a list.

Syntax

expression.ListTlgTag(ListType, ListArray, [ArchiveName], [Filter])

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListType (HMIGO_TLG_TAG_LIST_TYPE)</td>
<td>Defines which content should be returned in a list.</td>
</tr>
<tr>
<td></td>
<td>Possibilities are:</td>
</tr>
<tr>
<td></td>
<td>TLG_TG_NAMES (1) All tags created in a Tag Logging archive</td>
</tr>
<tr>
<td>ListArray (Variant)</td>
<td>List with the requested content.</td>
</tr>
<tr>
<td>ArchiveName (String)</td>
<td>Name of the archive in Tag Logging (optional). If the name of the archive is not specified, all archive tags are returned.</td>
</tr>
<tr>
<td>Filter (String)</td>
<td>Filters can be set optionally. Wildcards &quot;*&quot; and &quot;?&quot; are also possible.</td>
</tr>
</tbody>
</table>
Example:

In the following example, a check is made whether the archive tags are configured in the "Process Archive":

```
Sub ReadTlgTag()
    'HMIGO_029
    'read content in tag logging
    'no tags within archives are implemented
    Dim objHMIGO As New HMIGO
    Dim varRange As Variant
    Dim strArchive as String
    'set tlg archive name
    strArchive = "processarchive"
    'read all tlg tags in specified archive
    objHMIGO.ListTlgTag TLG_TAG_NAMES, arrContent, strArchive
    'check result
    If (UBound(arrContent) - LBound(arrContent) + 1) <= 0 Then
        MsgBox "no entries because no tag logging tags in specified archive are implemented"
    End If
End Sub
```

See also

- ListTlgArchive Function (Page 2604)
- GetTlgArchive Function (Page 2601)
- DeleteTlgTag Function (Page 2599)
- DeleteTlgArchive Function (Page 2598)
- CreateTlgTag Function (Page 2592)
- CreateTlgArchive Function (Page 2589)
- CommitTlgTag Function (Page 2587)
- CommitTlgArchive Function (Page 2585)
- CloseTlgTag Function (Page 2583)
- CloseTlgArchive Function (Page 2581)
- VBA in Tag Logging (Page 2573)

ListTlgTrigger function

Description

The function returns all created triggers in a list.
Syntax

Expression.ListTlgTrigger(ListType,ListArray,[Filter])

Expression

Required. An expression which returns a "HMIGO" type object.

Parameter

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
</table>
| ListType (HMIGO_TLG_TRIGGER_LIST_TYPE) | Defines the content that is to be returned in a list. Possibilities are:  
- TLG_TRIGGER_NAMES (1) all created triggers |
| ListArray (Variant) | List with the requested content. |
| Filter (String) | Filters can be set optionally. A trigger name can be used as a filter. Wildcards "*" and "?" are also possible. |

3.6.2.4 VBA in the Text Library

VBA in the Text Library

Introduction

VBA allows you to generate Text Library texts directly from the program code, modify and delete them, and display text IDs and texts.

Note

You should not have or should not open the "TextLibrary" editor when editing with VBA.

Principle

When you have created the instance of the "HMIGO" class, the following functions are available to you to access the TextLibrary:

- CreateTextLanguage
- CreateText
- DeleteText
- DeleteTextLanguage
- GetText
- GetTextID
The following enumerations are available for the parameter supply of these functions:

- HMIGO_TEXT_CREATE_MODE
- HMIGO_TEXT_LIST_TYPE

See also

- ModifyText Function (Page 2620)
- ListText Function (Page 2619)
- GetTextID Function (Page 2616)
- GetText Function (Page 2615)
- DeleteTextLanguage Function (Page 2614)
- DeleteText Function (Page 2612)
- CreateText Function (Page 2610)
- CreateTextLanguage Function (Page 2609)
- VBA in Other WinCC Editors (Page 2560)

CreateTextLanguage Function

Description

Creates a language in the Text Library.

Syntax

Expression.CreateTextLanguage(LanguageID)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID for the language to be created (e.g. 1031 for German, 1033 for English, etc.)</td>
</tr>
<tr>
<td></td>
<td>For a table of all language codes refer to the WinCC online help on &quot;Language Identifiers&quot;.</td>
</tr>
</tbody>
</table>
Example:

Sub CreateTextLanguage()
   ' HMIGO_016
   ' procedure to create a language in text library
   ' language must not be created before
   ' LanguageID german = 1031
   ' LanguageID english(US) = 1033
   ' LanguageID spanish = 1034
   ' LanguageID french = 1040
   ' LanguageID farsi= 1065
   ' declarations
   Dim objHMIGO As HMIGO
   Dim lngLanguageNumber As Long
   Set objHMIGO = New HMIGO
   lngLanguageNumber = 1065     'farsi
   'create new language
   objHMIGO.CreateTextLanguage lngLanguageNumber
   Set objHMIGO = Nothing
End Sub

See also

- ModifyText Function (Page 2620)
- ListText Function (Page 2619)
- GetTextID Function (Page 2616)
- GetText Function (Page 2615)
- DeleteTextLanguage Function (Page 2614)
- DeleteText Function (Page 2612)
- CreateText Function (Page 2610)
- VBA in the Text Library (Page 2608)

CreateText Function

Description

Creates a new text for the language specified. Text input for other languages can be added using ModifyText.

Syntax

Expression.CreateText(LanguageID,Text,CreateMode,TextID)

Expression

Necessary. An expression which returns a "HMIGO" type object.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID of the language for which the text will be created.</td>
</tr>
<tr>
<td>Text (string)</td>
<td>Text to be created.</td>
</tr>
<tr>
<td>CreateMode (HMIGO_TEXT_CREATE_MODE)</td>
<td>Mode of text creation:</td>
</tr>
<tr>
<td></td>
<td>● TEXT_ADD_REFCOUNT (0) only increases the reference counter when an identical text already exists.</td>
</tr>
<tr>
<td></td>
<td>● TEXT_CREATE_ALWAYS (1) always sets up a new text line and inserts the text in it.</td>
</tr>
<tr>
<td>TextID (long)</td>
<td>Returns the TextID assigned to the new text or the TextID whose reference counter is increased. This ID is required for processing the text in other functions.</td>
</tr>
</tbody>
</table>

Example:

Sub CreateText()
  ' HMIGO_017
  ' procedure to create a new text
  ' declarations
  Dim objHMIGO As HMIGO
  Dim lngLanguageID As Long
  Dim lngTextCreateMode As Long
  Dim lngTextID As Long 'return value of ".CreateText"
  Dim strText As String
  'LanguageID = english
  lngLanguageID = 1033
  '"TEXT_ADD_REFCOUNT" check if text exists, if not create new text
  lngTextCreateMode = 0
  '"TEXT_CREATE_ALWAYS" create always a new text (for messages)
  lngTextCreateMode = 1
  ' create new text
  objHMIGO.CreateText lngLanguageID, strText, lngTextCreateMode, lngTextID
  'show TextID of created text
  MsgBox "TextID: " & lngTextID, vbOKOnly, "Result CreateText"
  Set objHMIGO = Nothing
End Sub

See also

ModifyText Function (Page 2620)
ListText Function (Page 2619)
GetTextID Function (Page 2616)
DeleteText Function

Description

Deletes a line of text. All the languages for the corresponding line of text and the line of text itself are deleted.

Syntax

Expression.DeleteText(TextID)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextID (long)</td>
<td>ID of the line of text to be deleted</td>
</tr>
</tbody>
</table>
Example:

Sub DeleteText()
' HMIGO_018
' procedure to delete a text
' text will be searched and deleted
' declarations
Dim objHMIGO As HMIGO
Dim lngLanguageID As Long
Dim lngTextID As Long 'return value of GetTextID
Dim strText As String
On Error GoTo ErrorHandler
Set objHMIGO = New HMIGO
strText = "new text"
lngLanguageID = 1033

'first: find text in text library and return TextID
objHMIGO.GetTextID 1033, strText, lngTextID

'if searched text exists: delete this text
If Not lngTextID = -1 Then
    objHMIGO.DeleteText lngTextID
    MsgBox "Text : " & strText & " found in TextID: " & lngTextID & vbNewLine & _
    "TextID is deleted!", vbOKOnly, "Result DeleteText"
Else
    MsgBox "Text : " & strText & " not found." & vbNewLine & _
    "No Text deleted!", vbOKOnly, "Result DeleteText"
End If
Set objHMIGO = Nothing
Exit Sub
ErrorHandler:
'if lngText = (-1),searched text does not exit
If lngTextID = -1 Then
    'reset errorhandler
    Err.Clear
    Resume Next
End If
MsgBox "ErrNr. : " & Err.Number & vbNewLine & _
    "ErrDes.: " & Err.Description, vbOKOnly, "Error occurred"
    'reset errorhandler
    Err.Clear
End Sub

See also

VBA in the Text Library (Page 2608)
ModifyText Function (Page 2620)
ListText Function (Page 2619)
GetTextID Function (Page 2616)
GetText Function (Page 2615)
DeleteTextLanguage Function

Description

Enables a language to be deleted from the TextLibrary. In this case, all the texts in this language are also deleted.

Syntax

Expression.DeleteTextLanguage(LanguageID)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID of the language to be deleted</td>
</tr>
</tbody>
</table>

Example:

In the following example, a check is made whether the language '1036' exists. If it does, it will be deleted.

Sub DeleteLanguage()
    'HMIGO_030
    ' delete an existing languages in TextLibrary
    ' language '1036'/spanish has to exist
    Dim objHMIGO As New HMIGO
    Dim varRange As Variant
    Dim intLanguage As Long
    Dim lngPointer As Long
    ' get all existing languages
    objHMIGO.ListText TEXT_LANGUAGE_IDS, arrContent
    ' check requested list for language '1036'/ spanish and delete
    For lngPointer = LBound(arrContent) To UBound(arrContent)
        intLanguage = arrContent(lngPointer) + Val("&H400")
        If intLanguage = 1036 Then
            'delete language
            objHMIGO.DeleteTextLanguage intLanguage
        End If
    Next lngPointer
End Sub
See also

GetText Function (Page 2615)
ModifyText Function (Page 2620)
ListText Function (Page 2619)
GetTextID Function (Page 2616)
DeleteText Function (Page 2612)
CreateText Function (Page 2610)
CreateTextLanguage Function (Page 2609)
VBA in the Text Library (Page 2608)

GetText Function

Description

Returns the text for the selected text ID in the selected language.

Syntax

Expression.GetText(LanguageID, TextID, Text)

Expression

Necessary. An expression that returns an object of the type "HMIGeneralObjects".

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID of the language of the text to be read</td>
</tr>
<tr>
<td>TextID (long)</td>
<td>ID of the line of text from which text is to be read</td>
</tr>
<tr>
<td>Text (string)</td>
<td>Returns the text of the selected line of text and language.</td>
</tr>
</tbody>
</table>
Example:

Sub GetText()
    ' HMIGO_019
    ' procedure to get a text
    ' text with TextID = '69' need to be created
    ' declarations
    Dim objHMIGO As HMIGO
    Dim lngLanguageID As Long
    Dim lngTextID As Long
    Dim strText As String         'return value of GetText
    Set objHMIGO = New HMIGO
    lngTextID = 69
    lngLanguageID = 1033

    'find text text library
    objHMIGO.GetText lngLanguageID, lngTextID, strText

    'show found text
    MsgBox "Read Text in TextID : " & lngTextID & " is " & strText & " !", vbOKOnly, "Result GetText"
    Set objHMIGO = Nothing
End Sub

See also

ModifyText Function (Page 2620)
ListText Function (Page 2619)
GetTextID Function (Page 2616)
DeleteTextLanguage Function (Page 2614)
DeleteText Function (Page 2612)
CreateText Function (Page 2610)
CreateTextLanguage Function (Page 2609)
VBA in the Text Library (Page 2608)

GetTextID Function

Description

Returns the ID of the text searched for in the selected language.

If there are several texts with the same contents, only the line of text with the lowest ID is returned. Whether there are several lines of text with the same contents depends on the CreateMode of the CreateText function.
**syntax**

`Expression.GetTextID(LanguageID, Text, TextID)`

**Expression**

Necessary. An expression which returns a "HMIGO" type object.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID of the language of the text searched for</td>
</tr>
<tr>
<td>Text (string)</td>
<td>The text searched for</td>
</tr>
<tr>
<td>TextID (long)</td>
<td>ID of the line of text in which the text searched for was found</td>
</tr>
</tbody>
</table>
Example:

Sub GetTextID()
    ' HMIGO_020
    ' procedure to search a TextID
    ' text will be searched and a TextID will be returned
    ' declarations
    Dim objHMIGO As HMIGO
    Dim lngLanguageID As Long
    Dim lngTextID As Long         'return value of GetTextID
    Dim strText As String
    On Error GoTo ErrorHandler
    Set objHMIGO = New HMIGO
    strText = "old text"
    lngLanguageID = 1033
    'first: find text in text library and return TextID
    objHMIGO.GetTextID 1033, strText, lngTextID
    'if searched text exists: delete this text
    If Not lngTextID = -1 Then
        MsgBox "Text : " & strText & "" & lngTextID, _
        vbOKOnly, "Result GetTextID"
    Else
        MsgBox "Text : " & strText & "" not found!", vbOKOnly, "Result GetTextID"
    End If
    Set objHMIGO = Nothing
    Exit Sub
ErrorHandler:
    'if lngText = (-1),searched text does not exit
    If lngTextID = -1 Then
        'reset errorhandler
        Err.Clear
        Resume Next
    End If
    MsgBox "ErrNr. : " & Err.Number & vbNewLine & _
    "ErrDes.: " & Err.Description, vbOKOnly, "Error occurred"
    'reset errorhandler
    Err.Clear
End Sub

See also

ModifyText Function (Page 2620)
ListText Function (Page 2619)
GetText Function (Page 2615)
DeleteTextLanguage Function (Page 2614)
DeleteText Function (Page 2612)
CreateText Function (Page 2610)
ListText Function

Description
Alternatively, the ListText function returns the following contents of the TextLibrary as a list:
- All languages created
- All text IDs
- All texts in a specific language

Syntax
```vba
Expression.ListText(ListType, pListArray, [LanguageID], [Filter])
```

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListType (HMIGO_TEXT_LIST_TYPE)</td>
<td>Defines which content should be returned as a list. Possibilities are:</td>
</tr>
<tr>
<td></td>
<td>TEXT_LANGUAGE_IDS (1) All the created languages. The result still has to be converted by adding 400hex.</td>
</tr>
<tr>
<td></td>
<td>TEXT_IDS (2) All text IDs.</td>
</tr>
<tr>
<td></td>
<td>TEXT_TEXTS (3) All texts in a language.</td>
</tr>
<tr>
<td>pListArray (Variant)</td>
<td>List with the requested content.</td>
</tr>
<tr>
<td>LanguageID (Long)</td>
<td>The language ID whose text is to be returned.</td>
</tr>
<tr>
<td>Filter (String)</td>
<td>Filters can be set optionally. Wildcards &quot;*&quot; and &quot;?&quot; are also possible.</td>
</tr>
</tbody>
</table>
Example:

In the following example, a check is made whether the list with the text of a language is empty because the language does not exist:

```
Sub ReadTextsByLanguage()
    'HMIGO_031
    'read content in textLibrary by language
    Dim objHMIGO As New HMIGO
    Dim varRange As Variant
    Dim intLanguage As Integer
    'set invalid language ID
    intLanguage = 1051 'language does not exist
    'read all texts
    objHMIGO.ListText TEXT_TEXTS, arrContent, intLanguage
    'check result
    If (UBound(arrContent) - LBound(arrContent) + 1) <= 0 Then
        MsgBox "no entries because wrong language selection"
    End If
End Sub
```

See also

ModifyText Function (Page 2620)
GetTextID Function (Page 2616)
GetText Function (Page 2615)
DeleteTextLanguage Function (Page 2614)
DeleteText Function (Page 2612)
CreateTextLanguage Function (Page 2609)
VBA in the Text Library (Page 2608)

ModifyText Function

Description

Modifies the text for the selected language with the ID specified.

Syntax

```
Expression.ModifyText(LanguageID, TextID, Text)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.
Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageID (Long)</td>
<td>ID of the language of the text to be changed.</td>
</tr>
<tr>
<td>TextID (long)</td>
<td>ID of the language of the text to be changed.</td>
</tr>
<tr>
<td>Text (string)</td>
<td>New text to be inserted.</td>
</tr>
</tbody>
</table>

Example:

Sub ModifyText()
' HMIGO_021
' procedure to modify a text
' text will be searched and replaced
' declarations
Dim objHMIGO As HMIGO
Dim lngLanguageID As Long
Dim lngTextID As Long         'return value of GetTextID
Dim strOldText As String
Dim strNewText As String
On Error GoTo ErrorHandler
Set objHMIGO = New HMIGO
strOldText = "old text"
strNewText = "new text"
lngLanguageID = 1033

'first: find text in text library and return TextID
objHMIGO.GetTextID 1033, strOldText, lngTextID

'if searched text exists: replace this text
If Not lngTextID = -1 Then
    objHMIGO.ModifyText lngLanguageID, lngTextID, strNewText
    MsgBox "Text : " & strOldText & "" & vbCrLf & "" & vbCrLf & strNewText & ":", vbOKOnly, "Result DeleteText"
Else
    MsgBox "Text : " & strOldText & "" & vbCrLf & "" & vbCrLf & vbCrLf & vbCrLf & vbCrLf & "No Replacements done!", vbOKOnly, "Result DeleteText"
End If
Set objHMIGO = Nothing
Exit Sub
ErrorHandler:
'if lngTextID = (-1),searched text does not exit
If lngTextID = -1 Then
    'reset errorhandler
    Err.Clear
    Resume Next
Resume Next
End If

MsgBox "ErrNr. : " & Err.Number & vbCrLf & "ErrDes.: " & Err.Description, vbOKOnly, "Error occurred"
' reset errorhandler
Err.Clear
End Sub
See also

ListText Function (Page 2619)
GetTextID Function (Page 2616)
GetText Function (Page 2615)
DeleteTextLanguage Function (Page 2614)
DeleteText Function (Page 2612)
CreateText Function (Page 2610)
CreateTextLanguage Function (Page 2609)
VBA in the Text Library (Page 2608)

3.6.2.5 VBA in Alarm Logging

VBA in Alarm Logging

Introduction

VBA allows you to create messages directly from the program code, modify them, and delete them.

Note

You should not have or should not open the "Alarm Logging" editor when editing with VBA.

Principle

When you have created the instance of the "HMIGO" class, the following functions are available to you to access Alarm Logging:

- CloseSingleAlarm
- CommitSingleAlarm
- CreateSingleAlarm
- DeleteSingleAlarm
- GetSingleAlarm
- ListSingleAlarm

The following enumerations are available for the parameter supply of these functions:

- HMIGO_SINGLE_ALARM_CLASS_IDS
- HMIGO_SINGLE_ALARM_LIST_TYPE
### Access to the Object Properties

You can also access the parameters of the above-mentioned functions directly in VBA by means of the following object properties:

<table>
<thead>
<tr>
<th>Object property</th>
<th>Description</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectStateSingleAlarm</td>
<td>Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under &quot;VBA in other WinCC Editors&quot;.</td>
<td>Yes/no</td>
</tr>
<tr>
<td>SingleAlarmMessageNumber</td>
<td>Number of the message</td>
<td>Yes/no</td>
</tr>
<tr>
<td>SingleAlarmAGNumber</td>
<td>AS Number</td>
<td>Yes/yes</td>
</tr>
<tr>
<td>SingleAlarmCPUNumber</td>
<td>CPU number of the AGs.</td>
<td>Yes/yes</td>
</tr>
</tbody>
</table>
| SingleAlarmClassID       | Message class of the message. Possible values of the Enum SINGLE_ALARM_CLASS_IDS:  
  - SINGLE_ALARM_ERROR (1)  
  - SINGLE_ALARM_CLASS_2 (2)  
  - SINGLE_ALARM_CLASS_3 (3)  
  - SINGLE_ALARM_CLASS_4 (4)  
  - SINGLE_ALARM_CLASS_5 (5)  
  - SINGLE_ALARM_CLASS_6 (6)  
  - SINGLE_ALARM_CLASS_7 (7)  
  - SINGLE_ALARM_CLASS_8 (8)  
  - SINGLE_ALARM_CLASS_9 (9)  
  - SINGLE_ALARM_CLASS_10 (10)  
  - SINGLE_ALARM_CLASS_11 (11)  
  - SINGLE_ALARM_CLASS_12 (12)  
  - SINGLE_ALARM_CLASS_13 (13)  
  - SINGLE_ALARM_CLASS_14 (14)  
  - SINGLE_ALARM_CLASS_15 (15)  
  - SINGLE_ALARM_CLASS_16 (16)  
  - SINGLE_ALARM_CLASS_SYSTEM_REQUIRE_ACKNOWLEDGEMENT (17)  
  - SINGLE_ALARM_CLASS_SYSTEM_WITHOUT_ACKNOWLEDGEMENT (18) | Yes/yes    |
| SingleAlarmMessageTypeID | Type ID of the message. The permissible values depend on the message class:  
  - Class 1: Values from 1 to 16  
  - Class 2: Values from 17 to 32  
  - Class 3: Values from 33 to 48  
  - ...  
  - Class 18: 273 and 274 | Yes/yes    |
| SingleAlarmTextXXID XX = 1...10 | The properties SingleAlarmText1ID to SingleAlarmText10ID exist for the user texts 1 to 10. | Yes/yes    |
Sub DeleteSingleAlarmTagNameProcessValue1()
    'HMIGO_033
    Dim objGO as HMIGO
    Dim var as Long
    var = 0
    Set objGO = new HMIGO
    'message 1 will be modified
    objGO.GetSingleAlarm 1
    objGO.SingleAlarmTagNameProcessValue1 = var
    objGO.CommitSingleAlarm
    Set objGO = nothing
End Sub

See also
ListSingleAlarm Function (Page 2633)
GetSingleAlarm Function (Page 2632)
DeleteSingleAlarm Function (Page 2631)
CreateSingleAlarm Function (Page 2628)
CloseSingleAlarm Function

Description
Closes the message which is open.

Note
Modified parameters are not saved. If the current value should be saved, execute the CommitSingleAlarm() function again.

Syntax
Expression.CloseSingleAlarm()

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters
--
Example:

Sub CloseSingleAlarm()
    ' HMIGO_22
    ' procedure to open a singlealarm
    ' message #100 need to be created before
    ' declarations
    Dim objHMIGO As HMIGO
    Dim lngMsgNumber As Long
    Set objHMIGO = New HMIGO
    lngMsgNumber = 100
    ' current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
    ' open a singlealarm
    objHMIGO.GetSingleAlarm lngMsgNumber
    ' current status is "OPENED"
    MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
    ' close a singlealarm
    objHMIGO.CloseSingleAlarm
    ' current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
    Set objHMIGO = Nothing
End Sub

See also

ListSingleAlarm Function (Page 2633)
GetSingleAlarm Function (Page 2632)
DeleteSingleAlarm Function (Page 2631)
CreateSingleAlarm Function (Page 2631)
CommitSingleAlarm Function (Page 2628)
VBA in Alarm Logging (Page 2622)

CommitSingleAlarm Function

Description

Writes the changed parameters of the open message to WinCC.

Note

To change further parameters after a CommitSingleAlarm call, write these changes to WinCC by calling the function again.
syntax

Expression.CommitSingleAlarm()

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Example:

Sub CommitSingleAlarm()
' HMIGO_023
' procedure to change a property of a singlealarm
' message #100 need to be created before
' declarations
Dim objHMIGO As HMIGO
Dim lngMsgNumber As Long
Dim lngMsgBitNumber As Long
Set objHMIGO = New HMIGO
lngMsgNumber = 100
lngMsgBitNumber = 10
'current status is "EMPTY"
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
'open a singlealarm
objHMIGO.GetSingleAlarm lngMsgNumber
'current status is "OPENED" for changes
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
'change a property
objHMIGO.SingleAlarmMessageBit = lngMsgBitNumber
'current status is "MODIFIED" for changes
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
'commit a single alarm
objHMIGO.CommitSingleAlarm
'current status is "OPENED"
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"

Set objHMIGO = Nothing
End Sub

See also

ListSingleAlarm Function (Page 2633)
GetSingleAlarm Function (Page 2632)
DeleteSingleAlarm Function (Page 2631)
CreateSingleAlarm Function (Page 2628)
CreateSingleAlarm Function

Description

Creates a new message.

Syntax

Expression.CreateSingleAlarm(MessageNumber, ClassID, MessageTypeID, Text1ID, MessageTagName, MessageBit)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageNumber (long)</td>
<td>Number of the message.</td>
</tr>
<tr>
<td></td>
<td>● If an unused message number is specified here, it is accepted.</td>
</tr>
<tr>
<td></td>
<td>● If the value &quot;0&quot; is entered, the message number is freely assigned by the system. In this case, the message number is given back here.</td>
</tr>
<tr>
<td>ClassID (HMIGO_SINGLE_ALARM_CLASS_IDS)</td>
<td>Message class. The possible values are in the table Enum HMIGO_SINGLE_ALARM_CLASS_IDS.</td>
</tr>
<tr>
<td>MessageTypeID (Integer)</td>
<td>The permissible values depend on the message class:</td>
</tr>
<tr>
<td></td>
<td>● Class 1: Values from 1 to 16</td>
</tr>
<tr>
<td></td>
<td>● Class 2: Values from 17 to 32</td>
</tr>
<tr>
<td></td>
<td>● Class 3: Values from 33 to 48</td>
</tr>
<tr>
<td></td>
<td>● ...</td>
</tr>
<tr>
<td></td>
<td>● Class 18: Values from 263 to 288</td>
</tr>
<tr>
<td>Text1ID (Long)</td>
<td>ID for the first user text. The ModifySingleAlarm function can be used to define nine further user texts (1-10).</td>
</tr>
<tr>
<td>MessageTagName (String)</td>
<td>Tag name for the event.</td>
</tr>
<tr>
<td>MessageBit (integer)</td>
<td>Bit in bit reporting process (0...31)</td>
</tr>
</tbody>
</table>
Default Values When a New Message Is Created

The following table indicates the default values that are entered when a new message is created. These properties can be modified. The modifications are saved using the ModifySingleAlarm function.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Default Value (Enum Name =&gt; Value)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleAlarmAGNumber</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmCPUNumber</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmTextXXID</td>
<td>No text entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmTagNameProcessValueXX</td>
<td>No tag entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmQuitTag</td>
<td>No tag entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmQuitBits</td>
<td>0</td>
<td>No bits set.</td>
</tr>
<tr>
<td>SingleAlarmStateTag</td>
<td>No tag entered</td>
<td>Corresponds to exactly one day. Only relevant in the case of compressed tags.</td>
</tr>
<tr>
<td>SingleAlarmStateBits</td>
<td>0</td>
<td>No bits set.</td>
</tr>
<tr>
<td>SingleAlarmNormDLL</td>
<td>No name entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmQuitSingle</td>
<td>FALSE</td>
<td>Single acknowledgment</td>
</tr>
<tr>
<td>SingleAlarmHornActivate</td>
<td>FALSE</td>
<td>Horn Not active.</td>
</tr>
<tr>
<td>SingleAlarmArchiving</td>
<td>TRUE</td>
<td>Message will be archived.</td>
</tr>
<tr>
<td>SingleAlarmProtocol</td>
<td>TRUE</td>
<td>Message is logged.</td>
</tr>
<tr>
<td>SingleAlarmFlankInvert</td>
<td>FALSE</td>
<td>Not activated.</td>
</tr>
<tr>
<td>SingleAlarmLockedOnStart</td>
<td>FALSE</td>
<td>Message is not disabled.</td>
</tr>
<tr>
<td>SingleAlarmGlobalAPIFunction</td>
<td>FALSE</td>
<td>Message is not forwarded.</td>
</tr>
<tr>
<td>SingleAlarmActionName</td>
<td>No name entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmActionParams</td>
<td>No parameters entered for the action</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmInfoText</td>
<td>No text entered</td>
<td>--</td>
</tr>
<tr>
<td>SingleAlarmGroup</td>
<td>No text entered</td>
<td>--</td>
</tr>
</tbody>
</table>

Enum HMIGO_SINGLE_ALARM_CLASS_IDS

The following message classes are available for selection:

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE_ALARM_ERROR (1)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_2 (2)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_3 (3)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_4 (4)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_5 (5)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_6 (6)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_7 (7)</td>
<td>--</td>
</tr>
<tr>
<td>SINGLE_ALARM_CLASS_8 (8)</td>
<td>--</td>
</tr>
</tbody>
</table>
### Example:

```vba
Sub CreateSingleAlarm()
    ' HMIGO_024
    ' procedure to create a SingleAlarm
    ' message must not be created before
    ' message Text ID need to be created before in text library
    ' declarations
    Dim objHMIGO As HMIGO
    Dim strMsgText As String      'message text
    Dim strMsgTagName As String   'message variable
    Dim lngMsgNumber As Long      'message number
    Dim lngMsgBitNumber As Long   'bit number within the message variable
    Dim lngMsgTypeID As Long      'message type
    Dim lngMsgClassID             'SINGLE_ALARM_ERROR
    Dim lngMsgTextID As Long      'message text ID from textlibrary
    Set objHMIGO = New HMIGO
    strMsgText = "NewText"
    'current status is "EMPTY"
    MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
    'preset required parameter
    lngMsgNumber = 50
    lngMsgClassID = 1
    lngMsgTypeID = 2
    lngMsgTextID = 69
    strMsgTagName = "NewVariable"
    lngMsgBitNumber = 5
    'create a tag
    objHMIGO.CreateSingleAlarm lngMsgNumber, SINGLE_ALARM_ERROR, lngMsgTypeID, lngMsgTextID,
    strMsgTagName, lngMsgBitNumber
    'current status is "OPENED"
    MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
    Set objHMIGO = Nothing
End Sub
```
See also
ListSingleAlarm Function (Page 2633)
GetSingleAlarm Function (Page 2632)
DeleteSingleAlarm Function (Page 2631)
CommitSingleAlarm Function (Page 2626)
CloseSingleAlarm Function (Page 2625)
VBA in Alarm Logging (Page 2622)

DeleteSingleAlarm Function

Description
Deletes the specified message.

Syntax
Expression.DeleteSingleAlarm(MessageNumber)

Expression
Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageNumber (long)</td>
<td>Number of the message to be deleted.</td>
</tr>
</tbody>
</table>

Example:

Sub DeleteSingleAlarm()
' HMIGO_025
' procedure to delete a singlealarm
' message #100 need to be created before
' declarations
  Dim objHMIGO As HMIGO
  Dim lngMsgNumber As Long

  Set objHMIGO = New HMIGO
  lngMsgNumber = 100
  'current status is "EMPTY"
  MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"

  'delete a singlealarm
  objHMIGO.DeleteSingleAlarm lngMsgNumber
  Set objHMIGO = Nothing
End Sub
GetSingleAlarm Function

Description

Reads in the parameters of the message entered.
You can change or read the parameters by means of the object properties. You will find a list of the available object properties in this documentation under "VBA in Alarm Logging".

Syntax

Expression.GetSingleAlarm(MessageNumber)

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageNumber (long)</td>
<td>The message number of the message to be read in.</td>
</tr>
</tbody>
</table>
Example:

Sub GetSingleAlarm()
' HMIGO_026
' procedure to open a single alarm
' message #100 need to be created before
' declarations
Dim objHMIGO As HMIGO
Dim lngMsgNumber As Long
Set objHMIGO = New HMIGO
lngMsgNumber = 100
' current status is "EMPTY"
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
' open/ get a tag
objHMIGO.GetSingleAlarm lngMsgNumber
' current status is "OPENED"
MsgBox objHMIGO.ObjectStateSingleAlarm, vbOKOnly, "Status SingleAlarm"
Set objHMIGO = Nothing
End Sub

See also
ListSingleAlarm Function (Page 2633)
DeleteSingleAlarm Function (Page 2631)
CreateSingleAlarm Function (Page 2628)
CommitSingleAlarm Function (Page 2626)
CloseSingleAlarm Function (Page 2625)
VBA in Alarm Logging (Page 2622)

ListSingleAlarm Function

Description

The ListSingleAlarm function returns the content of Alarm Logging in a list:

- All actions created which are linked to messages
- All message class IDs created
- All info texts created
- All message numbers created
- All message type IDs created
- All message classes created
- All group messages created
syntax

Expression.ListSingleAlarm(ListType, pListArray, [Filter])

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

<table>
<thead>
<tr>
<th>Parameter (Data Type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListType (HMIGO_SINGLE_ALARM_LIST_TYPE)</td>
<td>Defines which content should be returned in a list. Possibilities are:</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_ACTION_NAMES (1) All actions created for Loop In Alarm when the parameter is set in the configuration as a string</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_CLASS_IDS (2) All message class IDs created</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_INFO_TEXTS (3) All info texts created</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_MESSAGE_NUMBERS (4) All message numbers created</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_MESSAGE_TYPE_IDS (5) All message type IDs created</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_GROUP_MESSAGE_CLASSES (6) All message classes created</td>
</tr>
<tr>
<td></td>
<td>● SINGLE_ALARM_GROUP_MESSAGE_USER_DEFINED (7) All group messages created</td>
</tr>
<tr>
<td>pListArray (Variant)</td>
<td>List with the requested content.</td>
</tr>
<tr>
<td>Filter (String)</td>
<td>Filters can be set optionally. Wildcards &quot;*&quot; and &quot;?&quot; are also possible.</td>
</tr>
</tbody>
</table>

Example:

In the following example, a check is made whether info texts have been configured:

Sub ReadSingleAlarm()
'HMIGO_032
'\read content in alarm logging
'\no info texts are implemented
  Dim objHMIGO As New HMIGO
Dim varRange As Variant
'\read all info texts
  objHMIGO.ListSingleAlarm SINGLE_ALARM_INFO_TEXTS, arrContent
'\check result
  If (UBound(arrContent) - LBound(arrContent) + 1) <= 0 Then
    MsgBox "no entries because no info texts are implemented"
End If
End Sub
See also

CreateSingleAlarm Function (Page 2628)
CommitSingleAlarm Function (Page 2626)
CloseSingleAlarm Function (Page 2625)
VBA in Alarm Logging (Page 2622)
Index

Adding,
DeactivateRTProject,
InquireLanguage,
TlgTrendWindowPressReportSaveButton

A

abort, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
abs, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
AcknowledgeMessage, 938, 939
acos, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
Action configuring, 1699
  with VBA, 1699
Action icon
  Features, 905
Action:Corrective measure, 63
Action:Delete, 35
Action:Editing, 57
Action:new creation, 56
Action:Save, 63
Action:Trigger, 18
ActionType property, 2107
Activating, 78
  of global actions in Runtime, 78
ActiveX controls, 1671
  Inserting with VBA, 1671
AddIn, 1734
  AddIn Manager, 1734
  create (example), 1738
  create with VB (example), 1739
  load automatically, 1734
  load manually, 1734
  Loading (instructions), 1737
  Loading behavior, 1734
Configuring with VBA, 1699
Create, 52
Creating functions and actions, 864, 905
Creating new actions, 909
CrossReference, 52
Defining, 864
Deleting, 887
Dialog "Info", 60, 62
Differences between actions and functions, 905
Edit, 52
Editing, 910, 932
Exporting, 922
Finding, 905
Importing, 923
Printing, 890
Protecting with a Password, 62
Protection against read and write access, 912
Renaming, 905, 924
Runtime behavior, 927
Saving, 913
Structuring, 864
Trigger:Configuring type "Timer", 70
Trigger:Configuring type Tag, 72
Use of DLLs, 876
Using actions from outside the project, 925
WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA

2637
unload, 1734
Use in Graphics Designer, 1734
Adding Timer trigger,
adding to picture, 82
Adding trigger, 918
Addins property, 2113
AdvancedAnalogDisplay, 1901
AdvancedStateDisplay, 1905
Aktion:Renaming, 76
Aktion:use multiple times, 20
Alarm Control, 286
Alarm Logging, 2622
Creating message with VBA, 2622
Deleting message with VBA, 2622
Editing message with VBA, 2622
Modifying message with VBA, 2622
Alarm object, 110
AlarmControl, 241
VBS example, 853
AlarmGoneVisible, 2114
AlarmLogs Object, 112
Alarms object (list), 110
Alarm object, 92
Animation trigger, 68
Application-specific menu, 1627
Add menu entry, 1631
Assigning help text, 1639
Assigning status text, 1639
Assigning VBA macro, 1642
Configuring, 1626
Creating, 1629
creating multiple languages, 1633
Application-specific toolbar, 1627
Add icon, 1637
Assigning help text, 1639
Assigning status text, 1639
Assigning VBA macro, 1642
Configuring, 1626
Creating, 1635
Archive backup: Connect via VBS, 838
asctime, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
asin, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
atoi, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095 atol, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095 AttachDB, 715
Attribute:GSC Runtime, 86
Authorization
Assigning, 922
Authorizations, 922
AXC_OnBtnAlarmHidingList, 947
AXC_OnBtnArcLong, 948
AXC_OnBtnArcShort, 949
AXC_OnBtnComment, 949
AXC_OnBtnEmergAckn, 950
AXC_OnBtnHideDlg, 951
AXC_OnBtnHideUnhideMsg, 952
AXC_OnBtnHit, 952
AXC_OnBtnHornAckn, 953
AXC_OnBtnInfo, 954
AXC_OnBtnLock, 955
AXC_OnBtnLockUnlock, 955
AXC_OnBtnLockWin, 956
AXC_OnBtnLoop, 957
AXC_OnBtnMsgFirst, 958
AXC_OnBtnMsgLast, 958
AXC_OnBtnMsgNext, 959
AXC_OnBtnMsgPrev, 960
AXC_OnBtnMsgWin, 961
AXC_OnBtnPrint, 961
AXC_OnBtnProtocol, 962
AXC_OnBtnScroll, 963
AXC_OnBtnSelect, 964
AXC_OnBtnSingleAckn, 964
AXC_OnBtnSortDlg, 965
AXC_OnBtnTimeBase, 966
AXC_OnBtnVisibleAckn, 967
AXC_SetFilter, 967

B
BackColor_Alarm, 2134
BackFillColor, 2136
BackFillColor_OK, 2136
BackFillColorSimulation, 2137
BackFillStyle, 2137
BackFillStyle_OK, 2137
BackFillStyleSimulation, 2137
Backup: Connect via VBS, 838
BasePicture, 2145
basics, 861
from VBS,
BitPosition0, 2149
BitSelect0, 2150
Bookmark: delete in Debugger, 100
Bookmark: set in Debugger, 100
Bookmark: skip to, 100
Bookmarks, 100
Breakpoint, 98, 99
Breakpoint: Deleting, 99
Breakpoint: set in Debugger, 98
bsearch, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Button1MessageClasses, 2166

C
C action, 1703
Configuring on an event with VBA, 1703
C script, 1694
Dynamize property with VBA, 1694
Trigger with VBA, 1707
VB action with VBA, 1705
ConnectionPoints, 2198
ConnectorObjects property, 2199
ConnectorType, 2200
Contents, 11
Controls
  WinCC Alarm Control, 286
  WinCC MediaControl, 256
Controls:HMI Symbol Library, 239
Controls:WinCC Digital Analog Clock, 246
Controls:WinCC Function Trend Control, 288
Controls:WinCC FunctionTrendControl, 248
Controls:WinCC Online Trend Control, 295
Controls:WinCC OnlineTrendControl, 257
Controls:WinCC RulerControl, 272
Controls:WinCC Slider Control, WinCC:WinCC Slider Control, 277
Controls:WinCC UserArchiveControl, 280
Create, 52, 56
  Action, 52
Create:Action, 56
Create:Procedure, 39
CreateDynamicDialog, 1827
CreateTagSet, 717
CreateTigTrigger (VBA), 2596
Creating
  Procedures, 36
CopyPasteSettings property, 2201
CopyRows, 716
CQBackColorOff, 2203
CQBackFlash, 2203
CQTextColorOff, 2203
CQTextFlash, 2204
CQTextColorOff, 2204
CQTextFlash, 2204
cos, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
cosh, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ctime, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ctype, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Customized Object, 1682
  Access with VBA, 1682
  Deleting, 1683
  Editing with VBA, 1683
  Properties, 1682
  ungroup, 1683
CutRows, 718
Cyclic, 64

D
DataItem Object, 113
DataLogs-Objekt, 114
DataSet Object (List), 116
DataSetObj, 1943
Debugger, 89, 90, 92, 94, 96, 97, 98, 99, 100, 101, 102
  Activating in WinCC, 89
Opening after error message, 89
Opening automatically, 89
Starting for Global Script, 89
Starting for Graphics Runtime, 89
Debugger:Basic Principles, 90, 97
Debugger:Change Picture During Debug, 90
Debugger:Delete bookmark, 100
Debugger:Deleting Breakpoints, 99
Debugger:Determine property values, 101
Debugger:Determine tag values, 101
Debugger:Executing Script Commands, 102
Debugger:modify property values, 101
Debugger:Modify tag values, 101
Debugger:Name of actions in the script file, 94
Debugger:Processing Scripts Step-by-Step, 97
Debugger:select running script, 96
Debugger:Set bookmark, 100
Debugger:Setting Breakpoints, 98
Debugger:skip to bookmark, 100
Debugger:Structure of the Script Files, 92
Declaration area, 57
Declaration Area:of an Action, 57
Delete:Actions and Procedures, 35
DeleteRows, 719
DeleteTextLanguage, 2614
DeleteTlgTrigger function (VBA), 2600
Deleting
Triggers, 75
Design of Global Script, 27
Design tool, 864
DetachDB, 719
Diagnostics, 81, 82, 84, 85, 86, 87
Diagnostics:Attributes:GSC Diagnostics, 82
Diagnostics:Debugger, 87
Diagnostics:GSC Diagnostics, 81
Diagnostics:GSC Runtime, 84
Diagnostics:GSC Runtime Attributes, 86
Diagnostics:Inserting GSC Diagnostics Window into a Picture, 82
Diagnostics:Inserting GSC Runtime into a Picture, 85
Diagnostics:VBS:Diagnostics, 80
difftime, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Digital Analog Clock, 246
Digital Clock, 246
Direct connection, 1701
Configuring with VBA, 1701
Directio, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
DisablePerformanceWarnings, 2213
Display property, 2214
div, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
DotNetControl, 1954
DrawInsideFrame, 2218
 DropDownListStyle, 2218
 Dynamic dialog, 1691
 Dynamize property with VBA, 1691
 DynamicStateType property, 2219
 Dynamization, 1687
  Dynamic dialog with VBA, 1687
  of properties with VBA, 1687
  Scripts with VBA, 1687
  Tag connection with VBA, 1687
 Dynamizing, 1689, 1694
  Property with C script with VBA, 1694
Index

Property with Dynamic dialog with VBA, 1691
Property with Tag Connection, 1689
Property with VB script with VBA, 1697

E
Edit, 52, 719, 1657
Action, 52
Copy of a Picture with VBA, 1658
Layer with VBA, 1657
Edit window
   Working in the edit window, 880
Edit:Action, 57
Editing, 57
   Procedures, 36
Editing function, 882
   Editing functions with the keyboard, 881
   Editing functions with the mouse, 882
Editing Objects with VBA, 1660
Editing Window, 30
Editing window:Color coding, 30
Editing Window:Global Script, 30
Editor, 26, 27, 33
Editor toolbars, 33
Editor:Global Script, 26
Editor:VBS-Editor in the Graphics Designer, 26
EnableFlash, 2221
Error, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
   1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
   1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
   1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
   1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
   1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
   1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
   1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
   1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
   1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
   1090, 1091, 1092, 1093, 1094, 1095
Error codes, 459
Error Messages, 411, 820
Error types, 90
Error types:Logical error, 90
Error types:Runtime Error, 90
Error types:Syntax Error, 90
Error:syntax, 63
Event, 1709
   application-specific, 1709
   Forwarding, 1709
   picture-specific, 1709
Event Handling, 1619, 1709, 1745
   Activated event, 1752
   Forwarding of events, 1615
Overview, 1745
   prevent, 1709
   Shutting down, 1709
Event Handling VBA Events,
   Event-driven, 64
   EventName, 2225
   EventQuitMask, 2222
Events, 1745
   Activated, 1752
   BeforeClose, 1752
   BeforeDocumentClose, 1753
   BeforeDocumentSave, 1754
   BeforeHMIObjectDelete, 1755
   BeforeLibraryFolderDelete, 1756
   BeforeLibraryObjectDelete, 1757
   BeforeQuit, 1759
   BeforeSave, 1760
   BeforeVisibleFalse, 1760
   ConnectionEvent, 1761
   DataLanguageChanged, 1762
   DesktopLanguageChanged, 1763
   DocumentActivated, 1764
   DocumentCreated, 1765
   DocumentOpened, 1766
   DocumentPropertyChanged, 1768
   DocumentSaved, 1767
   HMIObjectAdded, 1768
   HMIObjectMoved, 1769
   HMIObjectPropertyChanged, 1770
   HMIObjectResized, 1771
   LibraryFolderRenamed, 1772
   LibraryObjectAdded, 1774
   LibraryObjectDeleted, 1773
   MenuItemClicked, 1775
   NewLibraryFolder, 1776
   NewLibraryObject, 1777
   Opened, 1778
   Saved, 1779
   see reference:Events,
   SelectionChanged, 1780
   Started, 1781
   ToolbartemClicked, 1782
   VBA Event Handling, 1745
   ViewCreated, 1783
   WindowStateChanged, 1784
Example for VBS, 823
Example in WinCC:Configuring a diagnostics output
   via Trace, 827
Example in WinCC:Configuring a language
   change, 825
Example in WinCC:Configuring a picture change
   globally, 826
Example in WinCC: Configuring a picture change via Property, 827
Example in WinCC: Defining the color of objects, 825
Examples, 823
Examples in WinCC, 824, 825, 826, 827, 828, 830, 833, 836, 838
  VBS in WinCC, 823
  Writing object properties, 833
Examples in WinCC: Connect database backup with VBS, 838
Examples in WinCC: Accessing objects in Graphics Designer, 824
Examples in WinCC: Deactivate Runtime, 826
Examples in WinCC: Read tag values, 830
Examples in WinCC: Start an Action on the Server (Logging Object), 836
Executing, 1619
  VBA Macros, 1619
  exit, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ExitWinCC, 1511
exp, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Export, 720
  Access with VBA, 1713
  log on, 1713
ExternalShapeInfo, 1968
F
fab, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
  Features, 905
  LockStatus, 2315
  LockText, 2316
  LockTextColor, 2317
  LongStrokesBold, 2318
  LongStrokesOnly, 2318
  LongStrokesSize, 2319
  LongStrokesTextEach, 2319
  Marker, 2321
  Max, 2322
  MaximizeButton, 2322
  MaxZoom, 2323
  MCGUBackColorOff, 2324
  MCGUBackColorOn, 2324
  MCGUBackFlash, 2325
  MCGUTextColorOff, 2326
  MCGUTextColorOn, 2326
  MCGUTextFlash, 2327
  MCKOBackColorOff, 2328
  MCKOBackColorOn, 2329
  MCKOBackFlash, 2329
  MCKOTextColorOff, 2330
  MCKOTextColorOn, 2331
free, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
freeopen, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
frexp, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
fsetpos, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ftell, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Function
Application, 892
Compiling, 888, 892, 901
Creating, 864, 892
Creating new function, 895
Defining, 864
Differences between actions and functions, 905
Editing, 892
Finding, 892
Functions from other sources, 903
Printing, 890
Protection against read and write access, 900
Renaming, 902
Saving, 901
Structuring, 864
Use of DLLs, 876
Using internal functions, 897
Function code, 896
Writing, 896
Function Trend Control, 288
Functions, 2571
CloseSingleAlarm (VBA), 2625
CloseTag (VBA), 2565
CloseTlgArchive (VBA), 2581
CloseTlgTag (VBA), 2583
CommitSingleAlarm (VBA), 2626
CommitTag (VBA), 2566
CommitTlgArchive (VBA), 2585
CommitTlgTag (VBA), 2587
CreateSingleAlarm (VBA), 2628
CreateTag (VBA), 2568
CreateText (VBA), 2610
CreateTextLanguage (VBA), 2609
CreateTlgArchive (VBA), 2589
CreateTlgTag (VBA), 2592
DeleteSingleAlarm (VBA), 2631
DeleteTag (VBA), 2570
DeleteText (VBA), 2612
DeleteTlgArchive (VBA), 2598
DeleteTlgTag (VBA), 2599
GetSingleAlarm (VBA), 2632
GetTag (VBA), 2571
GetText (VBA), 2615
GetTextID (VBA), 2616
GetTlgArchive (VBA), 2601
GetTlgTag (VBA), 2602
ListSingleAlarm (VBA), 2633
ListTag (VBA), 2572
ListText (VBA), 2619
ListTlgArchive (VBA), 2604
ListTlgTag (VBA), 2606
ModifyText (VBA), 2620
FunctionTrendControl, 248
fwrite, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095

G

Gauge Control, 254
GCreateMyOperationMsg, 938
GCS-Runtime, 84
GCS-Runtime:Attribute, 86
General Examples, 855, 856, 858, 859
General examples:Configuring a database connection with VBS, 856
General examples:Retrieve MS automation interfaces, 858
General examples:Starting external application, 859
Get Focus, 1128
GetActualPointLeft, 1136
GetActualPointTop, 1137
GetAdaptBorder, 1189
GetAdaptPicture, 1190
GetAdaptSize, 1191
GetAlarmHigh, 1160
GetAlarmLow, 1161
GetAlignment, 1096
GetAlignmentLeft, 1129
GetAlignmentTop, 1130
GetAssignments, 1150
GetAssumeOnExit, 1151
GetAssumeOnFull, 1151
GetAverage, 1191
GetAxisSection, 1097
GetBackBorderWidth, 1240
GetBackColor, 1104
GetBackColor2, 1105
GetBackColor3, 1105
GetBackColorBottom, 1106
GetBackColorTop, 1106
GetBackFlashColorOff, 1120
GetBackFlashColorOn, 1121
GetBasePicReferenced, 1232
GetBasePicTransColor, 1233
GetBasePicture, 1234
GetBasePicUseTransColor, 1234
GetBitNumber, 1152
GetBorderColor, 1108
GetBorderStyle, 1241
GetBorderEndStyle, 1240
GetBorderFlashColorOff, 1121
GetBorderFlashColorOn, 1122
GetBorderStyle, 1241
GetBorderWidth, 1241
GetBoxAlignment, 1242
GetBoxCount, 1137
GetButtonType, 1192
GetCaption, 1193
GetCheckAlarmHigh, 1161
GetCheckAlarmLow, 1162
GetCheckLimitHigh4, 1162
GetCheckLimitHigh5, 1163
GetCheckLimitLow4, 1164
GetCheckLimitLow5, 1165
GetCheckToleranceHigh, 1165
GetCheckToleranceLow, 1166
GetCaption, 1193
GetClearOnError, 1153
GetClearOnNew, 1153
GetCloseButton, 1193
GetColorAlarmHigh, 1168
Index

GetColorAlarmLow, 1169
GetColorBottom, 1110
GetColorChangeType, 1194
GetColorLimitHigh4, 1169
GetColorLimitHigh5, 1170
GetColorLimitLow4, 1170
GetColorLimitLow5, 1171
GetColorToleranceHigh, 1172
GetColorToleranceLow, 1172
GetColorTop, 1111
GetColorWarningHigh, 1173
GetColorWarningLow, 1173
GetColor, 720, 731, 738, 739, 797, 798, 812, 813
GetColumnCollection, 721
GetCursorControl, 1195
GetCursorMode, 1195
GetDataFormat, 1154
GetDirection, 1138
GetEditAtOnce, 1196
GetEndAngle, 1139
goingenv, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
GetExponent, 1097
GetExtendedOperation, 1197
GetFillColor, 1111
GetFilling, 1118
GetFillingIndex, 1119
GetFillStyle2, 1243
GetFlashBackColor, 1122
GetFlashBorderColor, 1123
GetFlashPicture, 1235
GetFlashForeColor, 1124
GetFlashPicReferenced, 1235
GetFlashPicTransColor, 1236
GetFlashPicture, 1237
GetFlashPicUseTransColor, 1237
GetFlashRateBackColor, 1125
GetFlashRateBorderColor, 1125
GetFlashRateFlashPic, 1238
GetFontBold, 1130
GetFontItalic, 1131
GetFontName, 1132
GetFontSize, 1132
GetForeUnderline, 1133
GetForeColor, 1112
GetForeFlashColorOff, 1126
GetForeFlashColorOn, 1127
GetGrid, 1139
GetGridColor, 1113
GetGridHeight, 1140
GetGridWidth, 1140
GetHeight, 1141
GetHiddenInput, 1154
GetHitlistColumn, 722
GetHitlistColumnCollection, 723
GetHotkey, 1198
GetHWDiag, 1002
GetHWDiagLevel, 1003
GetHysteresis, 1198
GetHysteresisRange, 1199
GetIndex, 1239
GetInputValueChar, 1155
GetInputValueDouble, 1156
GetItemBorderBackColor, 1113
GetItemBorderColor, 1114
GetItemBorderStyle, 1244
GetItemBorderWidth, 1245
GetKopFupAwl, 1005
GetKopFupAwlLevel, 1006
GetLanguage, 1512
GetLanguageSwitch, 1199
GetLastChange, 1200
GetLayer, 1135
GetLeft, 1141
GetLeftComma, 1098
GetLimitHigh4, 1174
GetLimitHigh5, 1175
GetLimitLow4, 1175
GetLimitLow5, 1176
GetLimitMax, 1176
GetLimitMin, 1177
GetLink, 1188
GetLinkedVariable, 942
GetListType, 1156
GetLocalPicture, 943
GetMax, 1201
GetMaximizeButton, 1201
GetMessageBlock, 724
Index

GetTagDWord, 1462
GetTagDWordState, 1425
GetTagDWordStateQC, 1444
GetTagDWordStateQWait, 1433
GetTagDWordStateWait, 1416
GetTagDWordWait, 1454
GetTagFloat, 1463
GetTagFloatState, 1426
GetTagFloatStateQC, 1445
GetTagFloatStateQWait, 1434
GetTagFloatStateWait, 1417
GetTagFloatWait, 1454
GetTagMultiStateQWait, 1435
GetTagMultiStateWait, 1417
GetTagMultiWait, 1455
GetTagPrefix, 1212
GetTagRaw, 1463
GetTagRawState, 1426
GetTagRawStateQC, 1446
GetTagRawStateQWait, 1436
GetTagRawStateWait, 1418
GetTagRawWait, 1455
GetTagSByte, 1464
GetTagSByteState, 1427
GetTagSByteStateQC, 1447
GetTagSByteStateQWait, 1437
GetTagSByteStateWait, 1419
GetTagSByteWait, 1456
GetTagSDWord, 1465
GetTagSDWordState, 1428
GetTagSDWordStateQC, 1448
GetTagSDWordStateQWait, 1438
GetTagSDWordStateWait, 1420
GetTagSDWordWait, 1457
GetTagSWord, 1465
GetTagSWordState, 1428
GetTagSWordStateQC, 1449
GetTagSWordStateQWait, 1438
GetTagSWordStateWait, 1421
GetTagSWordWait, 1457
GetTagValue, 1466
GetTagValueStateQC, 1449
GetTagValueStateQWait, 1439
GetTagValueWait, 1458
GetTagWord, 1467
GetTagWordState, 1429
GetTagWordStateQC, 1450
GetTagWordStateQWait, 1440
GetTagWordStateWait, 1421
GetTagWordWait, 1459
GetText, 1134
GetTimeAxis, 747

GetTimeAxisCollection, 748
GetTimeColumn, 750
GetTimeColumnCollection, 751
GetTlGTrigger (VBA), 2604
GetToggleButton, 1246
GetToleranceHigh, 1178
GetToleranceLow, 1179
GetToolBarButton, 752
GetToolBarButtonCollection, 753
GetTop, 1148
GetTrend, 754, 1212
GetTrendCollection, 755
GetTrendColor, 1116
GetTrendWindow, 757
GetTrendWindowCollection, 758
GetTypeAlarmHigh, 1180
GetTypeAlarmLow, 1180
GetTypeLimitHigh4, 1181
GetTypeLimitHigh5, 1182
GetTypeLimitLow4, 1182
GetTypeLimitLow5, 1183
GetTypeToleranceHigh, 1184
GetTypeToleranceLow, 1185
GetTypeWarningHigh, 1185
GetTypeWarningLow, 1186
GetUnSelBGColor, 1117
GetUnSelTextColor, 1117
GetUpdateCycle, 1213
GetValueAxis, 759
GetValueAxisCollection, 760
GetValueColumn, 761
GetValueColumnCollection, 762
GetVisible, 1214
GetWarningHigh, 1187
GetWarningLow, 1187
GetWidth, 1148
GetWindowBorder, 1214
GetWindowsStyle, 1246
GetXAxis, 764
GetXAxisCollection, 765
GetYAxis, 766
GetYAxisCollection, 767
GetZeroPoint, 1149
GetZeroPointValue, 1215
GetZoom, 1216
Global action, 872
Application, 872
Characteristics, 872
Global C tags, 874
Application, 874
Definition, 874
Global Script, 878
  Action icon, 905
  Action structure, 864
  Adding action-related information, 910
  Adding function-related information, 898
  Assigning authorization, 922
  Authorizations, 922
  Cause for non-execution of an action, 905
  Changing trigger, 920
  Characteristics of global actions, 872
  Characteristics of internal functions, 870
  Characteristics of local actions, 871
  Characteristics of project functions, 867
  Characteristics of standard functions, 868
  Color code, 880
  Compiling action, 905, 913
  Compiling function, 888, 892, 901
  Computer properties, 873
  Creating action, 864, 905
  Creating function, 864
  Creating internal function, 895
  Creating new actions, 909
  Creating new function, 895
  Defining action, 864
  Defining function, 864
  Definition of global C tags,
  Deleting action, 887
  Deleting project functions, 887
  Deleting standard functions, 887
  Deleting trigger, 921
  Design tool, 864
  Differences between actions and functions, 905
  Editing action, 905, 910, 932
  Editing function, 892
  Editing functions with the keyboard, 882
  Editing functions with the mouse, 882
  Effect of triggers on actions, 914
  Exporting action, 922
  Features, 905
  Finding actions, 905
  Finding functions, 892
  Function structure, 864
  Generating new header, 887
  GSC Diagnose toolbar,
  GSC Runtime, 928
  GSC Runtime attributes, 932
  Identification of actions in the navigation window, 914
  Import action, 923
  Including Global Script Runtime in the startup list of the project,
  Including GSC Runtime in a process picture, 931

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
Global Script: Deleting Actions and Procedures, 35
Global Script: Toolbars, 33
Global Script: Working in the Editing Window, 30
Global Tag, 24
Global tag: Use in VBS, 24
GMsFunction, 940
gmtime, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
GNQBackColorOff, 2264
GNQBackColorFlash, 2265
GNQTextColorOff, 2265
GNQTextColorFlash, 2265
Graphic object, 143
Graphic Object: Properties, 143
Graphic Object: Types in VBS, 143
Graphic object: WinCC Gauge Control, 254
Graphics Designer, 1623
@GLOBAL.PDT, 1615
@PROJECT.PDT, 1615
Access to objects, 1660
Access to the component library with VBA, 1646
Access to the symbol library with VBA, 1646
Adapting with VBA, 1621, 1623
Add menu entry to user-defined menu, 1631
Adding a New Icon to the used-defined Toolbar, 1637
Application-specific menu, 1627
Application-specific toolbar, 1627
Assigning VBA macro to user-defined menu, 1642
Assigning VBA macro to user-defined toolbar, 1642
Configure user-defined menu, 1626
Configure user-defined toolbar, 1626
Create an application-specific menu, 1629
Create toolbar, 1635
Creating Menus in Multiple Languages, 1633
Editing objects with VBA, 1660
Executing VBA macros, 1619
Exporting VBA code, 1618
Importing VBA code, 1618
Object model, 1745
Organizing the VBA code, 1615
Paste Object from the component library into a picture with VBA, 1652
Paste Object from the symbol library into a picture with VBA, 1652
picture-specific menu, 1627
picture-specific toolbar, 1627
Template file, 1615
User-defined menu, 1627
User-defined toolbar, 1627
Group Object
Editing existing objects, 1679
Editing with VBA, 1676
Removing an object with VBA, 1676
Group Objects, 1674
Access to object properties, 1674
Adding an object with VBA, 1676
basics, 1674
Creating with VBA, 1676
Deleting with VBA, 1676
Editing with VBA, 1676
Ungrouping with VBA, 1676
Grouping, 1674
Access to object properties, 1674
Adding an object with VBA, 1676
basics, 1674
Creating with VBA, 1676
Deleting with VBA, 1676
Editing existing objects, 1679
Editing with VBA, 1674, 1676
Removing an object with VBA, 1676
Ungrouping with VBA, 1676
GSC Diagnose
Including in process picture, 934
Toolbar,
GSC Diagnostics, 83
GSC Diagnostics Attributes, 82
GSC Diagnostics: inserting into a Picture, 85
GSC Diagnostics: Attributes, 82
GSC Diagnostics: Toolbar, 83
GSC Runtime
Attributes, 932
Including in process picture, 931
GSC Runtime: Inserting into a Picture, 85

H
Header, 887
Regenerating, 887
HideAlarm, 769
Highlight Syntax, 41, 57
HitlistColumn object, 222
HitlistRelTimeFactorType, 449
HMIGO class, 2560
  Application, 2560
  Error Handling, 2560
HMIOObjects, 1745
  see HMIObjects, see the lists
HMIRuntime Object, 118
HMIUdoObjects, 2272

I
Information, 45, 60
  Adding action-related information, 910
  Adding to action, 60
  Adding to module, 45
  Adding to procedure, 45
Insert:GSC Diagnostics, 82
Insert:GSC Runtime, 85
Intellisense, 41, 57
Internal function, 870
  Application, 870
  Characteristics, 870
isahum, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
isgraph, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
islower, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
isprint, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
isdigit, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
IsPublished property, 2281

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
Index

ispunct, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
isspace, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
isupper, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
isxdigit, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
Item Object, 119

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA

L
labs, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
Language, 458
Layer, 1657
Controlling the visibility with VBA, 1657
CS, 1657
Editing with VBA, 1657
RT, 1657
Layer Object, 120
Layer00FillColor ..., 2295
Layer00FillStyle ..., 2295
Layers Object (Listing), 121
LDAssignments, 2298
ldexp, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
LDFontsType property, 2299
ldiv, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
ListBox object, 2007
Lists
Actions, 1898
AnalogResultInfo, 1908

2655


Index

Log, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095

log10, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095

Logging Object, 122
Logical errors, 90
LoopInAlarm, 772
lpszPictureName, 937

M

malloc, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095

MaxIndex, 2322
Index

MediaControl, 256
memchr, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
memcmp, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
memcpy, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Menu, 1627, 1654
  Add menu entry, 1631
  application-specific, 1627
  Assigning help text, 1639
  Assigning status text, 1639
  Assigning VBA macro, 1642
  Configuring, 1626
  Creating, 1629
  creating multiple languages, 1633
  picture-specific, 1627, 1654
  Placement, 1627
  Properties, 1627
  User-defined, 1627
Menu bar, 878
Menus and toolbars
  VBScript, 12
MenuToolBarConfig, 2339
MessageBlock object, 223
MessageColumn object, 223
  Activate, 1785
  Add, 1786
  Add (AnalogResultInfos Listing), 1787

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA

2657
Add (CustomToolbars Listing), 1788
Add (Documents Listing), 1790
Add (GroupedObjects), 1791
Add (TagTriggers Listing), 1792
Add (Views Listing), 1793
AddAction, 1794
AddActiveXControl, 1795
AddFolder, 1798
AddFromClipboard, 1799
AddHMIObject, 1801
AddItem, 1802
AddOLEObject, 1797, 1803, 1806
AlignBottom, 1807
AlignLeft, 1808
AlignRight, 1809
AlignTop, 1810
ArrangeMinimizedWindows, 1811
BackwardOneLevel, 1811
BringToFront, 1813
CascadeWindows, 1814
CenterHorizontally, 1815
CenterVertically, 1816
CheckSyntax, 1817
Close, 1818
CloseAll, 1819
ConvertToScript, 1820
ConvertToVBScript, 1821
CopySelection, 1823
CopyToClipboard, 1824
CreateCustomizedObject, 1825
CreateDynamic, 1828
CreateGroup, 1829
Delete, 1831
DeleteAll, 1832
DeleteDynamic, 1833
Destroy, 1835
DuplicateSelection, 1836
EvenlySpaceHorizontally, 1837
EvenlySpaceVertically, 1838
Export, 1840
Find, 1841
FindByDisplayName, 1842
FlipHorizontally, 1843
FlipVertically, 1845
ForwardOneLevel, 1846
GetDeclutterObjectSize, 1830
GetItemByPath, 1847
InsertFromMenu, 1848
InsertMenu, 1850
InsertMenuItem, 1852
InsertSeparator, 1853
InsertSubmenu, 1854
InsertToolbarItem, 1856
IsCSLayerVisible, 1857
IsRTLayerVisible, 1858
Item, 770, 1859
ItemByLcid, 1861
LoadDefaultConfig, 1863
MoveOneLayerDown, 1863
MoveOneLayerUp, 1864
MoveSelection, 1865
Open, 1867
Overview, 1745
PasteClipboard, 1868
PrintDocument, 1869
PrintProjectDocumentation, 1870
Remove, 1871
Rotate, 1872
SameHeight, 1874
SameWidth, 1876
SameWidthAndHeight, 1877
Save, 1879
SaveAll, 1880
SaveAs, 1880
SaveDefaultConfig, 1881
SelectAll, 1882
SendToBack, 1883
SetCSLayerVisible, 1885
SetDeclutterObjectSize, 1886
SetOpenContext, 1886
SetRTLayerVisible, 1887
ShowPropertiesDialog, 1888
ShowSymbolLibraryDialog, 1889
ShowTagDialog, 1890
TileWindowsHorizontally, 1890
TileWindowsVertically, 1891
Ungroup, 1892
Methods in VBS
ActivateDynamic, 713
Create, 716
DeactivateDynamic, 718
Methods: Activate, 712
Methods: Add, 713
Methods: Read, 783
Methods: Refresh, 788
Methods: Remove, 789
Methods: RemoveAll, 792
Methods: Restore, 794
Methods: Stop, 810
Methods: Trace, 811
Methods: Write, 813
Microsoft Script Debugger, 87
Module, 36, 39
   Adding information, 45
   Dialog "Info", 45, 47
   Editing, 36
   Name, 36
   Protecting with a Password, 47
   Save, 48
Module:Create, 39
Module:Renaming, 50
MoveAxis, 772
MoveRuler, 772
MoveToFirst, 775
MoveToFirstLine, 775
MoveToFirstPage, 775
MoveToLast, 776
MoveToLastLine, 776
MoveToLastPage, 777
MoveToNext, 777
MoveToNextLine, 777
MoveToNextPage, 778
MoveToPrevious, 778
MoveToPreviousLine, 779
MoveToPreviousPage, 779
MultiLineEdit object, 2016

N
Names, 94
Names:of actions in VBScript Files, 94
Navigation window
   Identification of actions in the navigation window, 914
New, 39, 56
New:Action, 56
New:Procedure, 39
NextColumn, 779
NextTrend, 780
NibbleSelect property, 2345

O
Object model, 1745
   Lists, 107
   Methods, 710
   Objects, 107
Object model:Properties, 301
Object names, 94
Object Types
   Controls, 218
   Customized object, 298
Object Types:Group, 300
Objects, 1745
   3DBarGraph, 1894
   ActiveXControl, 1900
   AnalogResultInfo, 1907
   ApplicationWindow, 1911
   BarGraph, 1914
   BinaryResultInfo, 1918
   BitResultInfo, 1919
   Button, 1921
   CheckBox, 1924
   Circle, 1928
   CircularArc, 1931
   ConnectorPoints (Listing), 1936
   CustomizedObject, 1938
   DataLanguage, 1941
   DestLink, 1945
   DirectConnection, 1947
   Document, 1949
   DynamicDialog, 1955
   Ellipse, 1957
   EllipseArc, 1960
   EllipseSegment, 1963
   Event, 1966
   FolderItem, 1974
   GraphicObject, 1975
   Group, 1978
   GroupDisplay, 1980
   HMIObject, 1988
   IOField, 1992
   LanguageFont, 1996
for a module, 47
for a Procedure, 47
Password input, 900, 912
Paste, 82, 85
PasteRows, 781
Picture, 1658
   Editing copy with VBA, 1658
Picture-specific menu, 1655
   Assigning help text, 1639
   Assigning status text, 1639
   Assigning VBA macro, 1642
Configuring, 1626
Creating, 1655
creating multiple languages, 1633
Picture-specific toolbar, 1655
   Assigning status text, 1639
   Assigning VBA macro, 1642
Configuring, 1626
Creating, 1655
Pinnable property, 2378
Pinned property, 2378
pow, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
PreviousColumn, 781
PreviousTrend, 781
Print, 782
Print parameters, 890
   Setting, 890
Print:VBS, 104
PrioAlarm, 2383
PrioBit16, 2383
procedure, 1619
   Adding information, 45
   Creating, 36
   Dialog "Info", 45, 47
   Editing, 36
   Name, 36
   Protecting with a Password, 47
   Save, 48
   Version, 1619
Procedure:Change, 41
Procedure:Delete, 35
Procedure:New creation, 39
Procedure:Project Procedure, 44
Procedure:Rename, 50
Procedure:Standard procedure, 44
Procedure:Storage location in WinCC, 20
Procedure:use multiple times, 20
Procedure:Write Code, 41
Process, 2384
Process1, 2384
Process2, 2385
Process3, 2385
ProcessValue Object, 123
ProcessValues Object (List), 124
ProgramExecute, 1009
Project documentation
   Printing, 891
Project function, 867
   Application, 867
   Applying, 897
   Characteristics, 867
   Deleting, 887
Project language, 1624
Access with VBA, 1624
Project Object, 124
Project Procedure:Use, 44
Properties, 898, 900, 910, 912, 920, 921, 1745
   Actions, 2106
   ActiveDocument, 2107
   ActiveLayer, 2108
   ActualPointLeft, 2109
   ActualPointTop, 2110
   AdaptBorder, 2111
   AdaptPicture, 2111
   AdaptSize, 2112
   AlarmHigh, 2114
   AlarmLow, 2115
   Alignment, 2116
   AlignmentLeft, 2117
   AlignmentTop, 2118
   AnalogResultInfos, 2118
   AngleAlpha, 2119
   AngleBeta, 2120
   Application, 2121
   ApplicationDataPath, 2121
   Assignments, 2122
   AssumeOnExit, 2123
   AssumeOnFull, 2123
   AutomationName, 2124
   AvailableDataLanguages, 2127
   Average, 2128
   Axe, 2129
   AxisSection, 2130
   BackBorderWidth, 2131
Family, 2229
FillColor, 2231
Filling, 2232
FillingIndex, 2233
FillStyle, 2235
FillStyle2, 2237
FillStyleAlignment, 2239
FlashBackColor, 2239
FlashBorderColor, 2240
FlashForeColor, 2241
FlashRate, 2246
FlashRateBackColor, 2247
FlashRateBorderColor, 2248
FlashRateForeColor, 2249
FolderItems, 2254
FontBold, 2256
FontColor, 2260
FontFlashColorOff, 2261
FontFlashColorOn, 2262
FontItalic, 2263
FontSize, 2264
ForeColor, 2265
GlobalColorScheme, 2269
Grid, 2266
GridColor, 2268
GridHeight, 2269
GroupedHMIObjects, 2269
Height, 2269
HiddenInput, 2270
Hotkeys, 2272
Hysteresis, 2273
HysteresisRange, 2274
Icon, 2275, 2276, 2278
Index, 2276
InputValue, 2279
IsActive, 2280
IsConnectedToProject, 2280
IsDynamicable, 2280
Italic, 2282
Item, 2283
ItemBackColor, 2283
ItemBorderColor, 2284
ItemBorderStyle, 2285
ItemBorderWidth, 2286
InputValue, 2279
IsActive, 2280
IsConnectedToProject, 2280
IsDynamicable, 2280
Index

PicDeactReferenced, 2365
PicDeactTransparent, 2366
PicDeactUseTransColor, 2367
PicDownReferenced, 2367
PicDownTransparent, 2368
PicDownUseTransColor, 2369
PicReferenced, 2370
PicTransColor, 2371
PictureDeactivated, 2371
PictureDown, 2372
PictureName, 2373
PictureUp, 2374
PicUpReferenced, 2375
PicUpTransparent, 2376
PicUpUseTransColor, 2377
PicUseTransColor, 2377
PointCount, 2379
Position, 2380
PositiveValue, 2380
PredefinedAngels, 2381
Pressed, 2382
Process, 2383
ProfileName, 2385
ProgID, 2386
ProjectName, 2386
ProjectType, 2387
Properties, 2388
prototype, 2389
QualityCodeStateChecked, 2390
QualityCodeStateValues, 2392
Radius, 2394
RadiusHeight, 2394
RadiusWidth, 2395
RangeTo, 2396
ReferenceRotationLeft, 2396
ReferenceRotationTop, 2397
Relevant, 2398
ResultType, 2399
RightComma, 2399
RotationAngle, 2400
RoundCornerHeight, 2401
RoundCornerWidth, 2402
SameSize, 2403
ScaleColor, 2403
ScaleTicks, 2404
Scaling, 2405
ScalingMode, 2405
ScalingType, 2406
ScriptType, 2407
ScrollBars, 2408
ScrollPositionX, 2408
ScrollPositionY, 2409
ScrollPosX, 2411
ScrollPosY, 2411
see properties, 2412
SelBGColor, 2412
Selected, 2412
SelIndex, 2414
SelText, 2414
SelTextColor, 2414
ServerName, 2415
ServerPrefix, 2416
ShortCut, 2416
SignificantMask, 2419
Size, 2419
Sizeable, 2421
SmallChange, 2421
SnapToGrid, 2422
SourceCode, 2424
SourceLink, 2422
StartAngle, 2426
StatusText, 2426
SubMenu, 2427
SymbolLibraries, 2428
TabOrderAllHMIObjects, 2430
TabOrderAlpha, 2429
TabOrderKeyboard, 2431
TabOrderMouse, 2432
TabOrderOtherAction, 2432
TabOrderSwitch, 2433
Tag, 2434
TagPrefix, 2436
Text, 2439
Toggle, 2441
ToleranceHigh, 2442
ToleranceLow, 2442
ToolbarItems, 2443
ToolTipText, 2445
Top, 2446
TopConnectedObjectName, 2446
TopConnectedPointIndex, 2447
Transparency, 2447
Trend, 2448
TrendColor, 2449
Trigger, 2449
Type:, 2450
TypeAlarmHigh, 2451
TypeAlarmLow, 2452
TypeLimitHigh4, 2453
TypeLimitHigh5, 2453
TypeLimitLow4, 2454
TypeLimitLow5, 2455
TypeToleranceHigh, 2456
TypeToleranceLow, 2457
TypeWarningHigh, 2458
TypeWarningLow, 2459
Underlined, 2460
UnselBGColor, 2462
UnselTextColor, 2462
UpdateCycle, 2463
UserValue1, 2466
UserValue2, 2467
UserValue3, 2467
UserValue4, 2468
Value, 2469
VALUE_ACCESS_FAULT, 2470
VALUE_ADDRESS_ERROR, 2472
VALUE_BAD_COMM_LVL, 2474, 2476
VALUE_BAD_CONFERROR, 2478
VALUE_BAD_DEVICE, 2480
VALUE_BAD_MISCSTATES, 2482
VALUE_BAD_NOSPECIFIC, 2484
VALUE_BAD_NOT_CONNECTED, 2486
VALUE_BAD_OUTOFSEVR, 2488
VALUE_BAD_PROCRELNOM, 2490
VALUE_BAD_PROCREL_SUB, 2492
VALUE_CONVERSION_ERROR, 2494
VALUE_HANDSHAKE_ERROR, 2496
VALUE_HARDWARE_ERROR, 2498
VALUE_HIGHLIMITED, 2500
VALUE_INVALID_KEY, 2502
VALUE_LOWLIMITED, 2504
VALUE_MAX_LIMIT, 2506
VALUE_MAX_RANGE, 2508
VALUE_MIN_LIMIT, 2510
VALUE_MIN_RANGE, 2512
VALUE_NOT_ESTABLISHED, 2514
VALUE_SERVERDOWN, 2516
VALUE_STARTUP_VALUE, 2518
VALUE_TIMEOUT, 2520
VALUE_UNCERT_ENGVHIGHLIM, 2522
VALUE_UNCERT_ENGVLOWLIM, 2524
VALUE_UNCERT_ENGVONLIM, 2526
VALUE_UNCERT_INITVAL, 2528
VALUE_UNCERT_LUV, 2530
VALUE_UNCERT_MAINTDEMO, 2532
VALUE_UNCERT_MISCSTATES, 2534
VALUE_UNCERT_NOSPECIFIC, 2536
VALUE_UNCERT_PROCRELNOM, 2538
VALUE_UNCERT_SIMVAL, 2540
VALUE_UNCERT_SUBSTSET, 2542
VariablesExist, 2544
VariableStateChecked, 2545
VariableStateType, 2546
VariableStateValues, 2546
VariableTriggers, 2548
VarName, 2548
VBAVersion, 2549
VBE, 2550
Version, 2550
Views, 2550
Visible, 2551
WarningHigh, 2552
WarningLow, 2553
Width, 2553, 2554
WindowBorder, 2555
WindowMonitorNumber, 2556
WindowPositionMode, 2556
WindowsStyle, 2557
WindowState, 2557
ZeroPoint, 2558
ZeroPointValue, 2559
Zoom, 2559
Properties,
Properties in VBA
CommonVBSEventArea, 2195
CommonVBSPropertyArea, 2196
FaceplateType, 2228
TagScaleParam1, 2437
TagScaleParam2, 2437
TagScaleParam3, 2438
TagScaleParam4, 2438
Properties in VBS, 301
AlarmID, 308
BackStyle, 325
Comment, 390
ComputerName, 391
Context, 391
FillStyle, 419
FillStyleAlignment, 421
GlobalColorScheme, 438
HiddenInput, 311, 312, 445, 483, 516, 543, 577, 676, 702
IndependentWindow, 451
Index, 452
InputValue, 454
Instance, 454
ItemProviderClsid, 456
LineJoinStyle, 480
NumberLines, 511
ProcessValue, 545
RotationAngle, 552
SavedTrend, 557
Selected Trend, 565
SellIndex, 568
SelText, 568
Smart tag, 580
SortOrder, 580
Properties in VBS/ScrollPositionX, 562
Properties in VBS/ScrollPositionY, 562
Properties in VBS:
DesiredCurveSourceUAColumnY, 404
Properties in VBS:AccessPath, 302
Properties in VBS:Activate, 302
Properties in VBS:ActiveProject, 303
Properties in VBS:ActiveScreen, 304
Properties in VBS:ActiveScreenItem, 304
Properties in VBS:Actualize, 305
Properties in VBS:ActualPointLeft, 305
Properties in VBS:ActualPointTop, 306
Properties in VBS:AdaptBorder, 306
Properties in VBS:AdaptPicture, 307
Properties in VBS:AdaptSize, 307
Properties in VBS:AdjustRuler, 307
Properties in VBS:AlarmHigh, 308
Properties in VBS:AlarmLogs, 308
Properties in VBS:AlarmLow, 309
Properties in VBS:Alignment, 309
Properties in VBS:AlignmentLeft, 309
Properties in VBS:AlignmentTop, 310
Properties in VBS:AllowPersistence, 310
Properties in VBS:AllServer, 311
Properties in VBS:Analog, 311
Properties in VBS:AngleAlpha, 312
Properties in VBS:AngleBeta, 312
Properties in VBS:AngleMax, 312
Properties in VBS:AngleMin, 313
Properties in VBS:Application, 313
Properties in VBS:Archive, 314
Properties in VBS:Assignments, 315
Properties in VBS:AssumeOnExit, 315
Properties in VBS:AssumeOnFull, 315
Properties in VBS:AutoRange, 317
Properties in VBS:AutoRangeX, 317
Properties in VBS:AutoRangeY, 317
Properties in VBS:AutoScroll, 317
Properties in VBS:AutoSize, 319
Properties in VBS:Average, 320
Properties in VBS:Axe, 320
Properties in VBS:Axis, 602
Properties in VBS:AxisSection, 321
Properties in VBS:BackBorderWidth, 321
Properties in VBS:BackColor, 322
Properties in VBS:BackColor2, 323
Properties in VBS:BackColor3, 323
Properties in VBS:BackColorBottom, 323
Properties in VBS:BackColorTop, 323
Properties in VBS:BackFlashColorOff, 324
Properties in VBS:BackFlashColorOn, 324
Properties in VBS:Background, 324
Properties in VBS:Background Picture, 324
Properties in VBS:BackPictureAlignment, 325, 536
Properties in VBS:BackpictureName, 325
Properties in VBS:BarBackColor, 326
Properties in VBS:BarDepth, 328
Properties in VBS:BarFillColor, 329
Properties in VBS:BarHeight, 329
Properties in VBS:BarWidth, 340
Properties in VBS:BasePicReferenced, 340
Properties in VBS:BasePicTransColor, 341
Properties in VBS:BasePicture, 341
Properties in VBS:BasePicUseTransColor, 341
Properties in VBS:BaseScreenName, 342
Properties in VBS:BaseX, 343
Properties in VBS:BaseY, 342
Properties in VBS:BeginTime, 343
Properties in VBS:BeginValue, 343
Properties in VBS:BeginWidth, 346
Properties in VBS:BeginX, 344
Properties in VBS:BeginY, 344
Properties in VBS:BevelColorDown, 344
Properties in VBS:BevelColorUp, 345
Properties in VBS:BevelInner, 345
Properties in VBS:BevelOuter, 345
Properties in VBS:BitNumber, 346
Properties in VBS:BlinkColor, 346
Properties in VBS:BorderBackColor, 353
Properties in VBS:BorderColor, 353
Properties in VBS:BorderColorBottom, 354
Properties in VBS:BorderColorTop, 354
Properties in VBS:BorderEndStyle, 354
Properties in VBS:BorderFlashColorOff, 354
Properties in VBS:BorderFlashColorOn, 355
Properties in VBS:BorderStyle, 355
Properties in VBS:BottomConnectedConnectionPointIndex, 356
Properties in VBS:BottomConnectedObjectName, 356
Properties in VBS:BoxAlignment, 356
Properties in VBS:BoxCount, 357
Properties in VBS:BoxType, 357
Properties in VBS:Button1Width, 361
Properties in VBS:Button2Width, 361
Properties in VBS:Button3Width, 362
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties in VBS:Button4Width</td>
<td>362</td>
</tr>
<tr>
<td>Properties in VBS:ButtonColor</td>
<td>357</td>
</tr>
<tr>
<td>Properties in VBS:ButtonCommand</td>
<td>358</td>
</tr>
<tr>
<td>Properties in VBS:Caption</td>
<td>362</td>
</tr>
<tr>
<td>Properties in VBS:CaptionColor</td>
<td>363</td>
</tr>
<tr>
<td>Properties in VBS:CaptionFont</td>
<td>363</td>
</tr>
<tr>
<td>Properties in VBS:CaptionOffset</td>
<td>364</td>
</tr>
<tr>
<td>Properties in VBS:CaptionText</td>
<td>364</td>
</tr>
<tr>
<td>Properties in VBS:CellCut</td>
<td>364</td>
</tr>
<tr>
<td>Properties in VBS:CenterColor</td>
<td>366</td>
</tr>
<tr>
<td>Properties in VBS:CenterScale</td>
<td>366</td>
</tr>
<tr>
<td>Properties in VBS:CheckAlarmHigh</td>
<td>367</td>
</tr>
<tr>
<td>Properties in VBS:CheckAlarmLow</td>
<td>367</td>
</tr>
<tr>
<td>Properties in VBS:CheckLimitHigh4</td>
<td>367</td>
</tr>
<tr>
<td>Properties in VBS:CheckLimitHigh5</td>
<td>368</td>
</tr>
<tr>
<td>Properties in VBS:CheckLimitLow4</td>
<td>368</td>
</tr>
<tr>
<td>Properties in VBS:CheckLimitLow5</td>
<td>368</td>
</tr>
<tr>
<td>Properties in VBS:CheckToleranceHigh</td>
<td>369</td>
</tr>
<tr>
<td>Properties in VBS:CheckToleranceLow</td>
<td>369</td>
</tr>
<tr>
<td>Properties in VBS:CheckWarningHigh</td>
<td>369</td>
</tr>
<tr>
<td>Properties in VBS:CheckWarningLow</td>
<td>370</td>
</tr>
<tr>
<td>Properties in VBS:ClearOnError</td>
<td>370</td>
</tr>
<tr>
<td>Properties in VBS:ClearOnNew</td>
<td>370</td>
</tr>
<tr>
<td>Properties in VBS:Closeable</td>
<td>370</td>
</tr>
<tr>
<td>Properties in VBS:CloseButton</td>
<td>371</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGrid</td>
<td>371</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGridValue</td>
<td>372</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGridValueX</td>
<td>373</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGridValueY</td>
<td>373</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGridX</td>
<td>372</td>
</tr>
<tr>
<td>Properties in VBS:CoarseGridY</td>
<td>372</td>
</tr>
<tr>
<td>Properties in VBS:CollectValue</td>
<td>373</td>
</tr>
<tr>
<td>Properties in VBS:ColMove</td>
<td>374</td>
</tr>
<tr>
<td>Properties in VBS:Color</td>
<td>374</td>
</tr>
<tr>
<td>Properties in VBS:ColorAlarmHigh</td>
<td>374</td>
</tr>
<tr>
<td>Properties in VBS:ColorAlarmLow</td>
<td>375</td>
</tr>
<tr>
<td>Properties in VBS:ColorBottom</td>
<td>375</td>
</tr>
<tr>
<td>Properties in VBS:ColorChangeType</td>
<td>375</td>
</tr>
<tr>
<td>Properties in VBS:ColorLimitHigh4</td>
<td>376</td>
</tr>
<tr>
<td>Properties in VBS:ColorLimitHigh5</td>
<td>376</td>
</tr>
<tr>
<td>Properties in VBS:ColorLimitLow4</td>
<td>376</td>
</tr>
<tr>
<td>Properties in VBS:ColorLimitLow5</td>
<td>377</td>
</tr>
<tr>
<td>Properties in VBS:ColorToleranceHigh</td>
<td>377</td>
</tr>
<tr>
<td>Properties in VBS:ColorToleranceLow</td>
<td>377</td>
</tr>
<tr>
<td>Properties in VBS:ColorTop</td>
<td>377</td>
</tr>
<tr>
<td>Properties in VBS:ColorWarningHigh</td>
<td>378</td>
</tr>
<tr>
<td>Properties in VBS:ColorWarningLow</td>
<td>378</td>
</tr>
<tr>
<td>Properties in VBS:ColTitle</td>
<td>378</td>
</tr>
<tr>
<td>Properties in VBS:ColWidth</td>
<td>389</td>
</tr>
<tr>
<td>Properties in VBS:Command</td>
<td>390</td>
</tr>
<tr>
<td>Properties in VBS:CommonTime</td>
<td>390</td>
</tr>
<tr>
<td>Properties in VBS:CommonX</td>
<td>391</td>
</tr>
<tr>
<td>Properties in VBS:CommonY</td>
<td>391</td>
</tr>
<tr>
<td>Properties in VBS:ContinuousChange</td>
<td>393</td>
</tr>
<tr>
<td>Properties in VBS:Count</td>
<td>393</td>
</tr>
<tr>
<td>Properties in VBS:CurrentContext</td>
<td>394</td>
</tr>
<tr>
<td>Properties in VBS:Cursor</td>
<td>395</td>
</tr>
<tr>
<td>Properties in VBS:CursorControl</td>
<td>395</td>
</tr>
<tr>
<td>Properties in VBS:CurveForm</td>
<td>396</td>
</tr>
<tr>
<td>Properties in VBS:Danger</td>
<td>397</td>
</tr>
<tr>
<td>Properties in VBS:DangerColor</td>
<td>397</td>
</tr>
<tr>
<td>Properties in VBS:DataFormat</td>
<td>397</td>
</tr>
<tr>
<td>Properties in VBS:DataIndex</td>
<td>398</td>
</tr>
<tr>
<td>Properties in VBS:DataLogs</td>
<td>398</td>
</tr>
<tr>
<td>Properties in VBS:DataSet</td>
<td>398</td>
</tr>
<tr>
<td>Properties in VBS:DataX</td>
<td>399</td>
</tr>
<tr>
<td>Properties in VBS:DataXY</td>
<td>399</td>
</tr>
<tr>
<td>Properties in VBS:DataY</td>
<td>400</td>
</tr>
<tr>
<td>Properties in VBS:DeleteData</td>
<td>402</td>
</tr>
<tr>
<td>Properties in VBS:Delta</td>
<td>402</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveColor</td>
<td>402</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveCurveForm</td>
<td>403</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveSourceNumberOfUAValues</td>
<td>403</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveSourceUAArchive</td>
<td>403</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveSourceUAArchiveStartID</td>
<td>404</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveSourceUAColumnX</td>
<td>404</td>
</tr>
<tr>
<td>Properties in VBS:DesiredCurveVisible</td>
<td>405</td>
</tr>
<tr>
<td>Properties in VBS:Direction</td>
<td>405</td>
</tr>
<tr>
<td>Properties in VBS:DisplayOptions</td>
<td>406</td>
</tr>
<tr>
<td>Properties in VBS:Edit</td>
<td>407</td>
</tr>
<tr>
<td>Properties in VBS:Editable</td>
<td>407</td>
</tr>
<tr>
<td>Properties in VBS:EditAtOnce</td>
<td>407</td>
</tr>
<tr>
<td>Properties in VBS:Enabled</td>
<td>408</td>
</tr>
<tr>
<td>Properties in VBS:EndTime</td>
<td>409</td>
</tr>
<tr>
<td>Properties in VBS:EndAngle</td>
<td>409</td>
</tr>
<tr>
<td>Properties in VBS:EndTime</td>
<td>410</td>
</tr>
<tr>
<td>Properties in VBS:EndValue</td>
<td>410</td>
</tr>
<tr>
<td>Properties in VBS:EndX</td>
<td>411</td>
</tr>
<tr>
<td>Properties in VBS:EndY</td>
<td>411</td>
</tr>
<tr>
<td>Properties in VBS:ErrorDescription</td>
<td>411</td>
</tr>
<tr>
<td>Properties in VBS:Exponent</td>
<td>412</td>
</tr>
<tr>
<td>Properties in VBS:ExtendedOperation</td>
<td>415</td>
</tr>
<tr>
<td>Properties in VBS:ExtendedZoomingEnable</td>
<td>416</td>
</tr>
<tr>
<td>Properties in VBS:FillColor</td>
<td>418</td>
</tr>
<tr>
<td>Properties in VBS:Filling</td>
<td>419</td>
</tr>
<tr>
<td>Properties in VBS:FillingIndex</td>
<td>419</td>
</tr>
<tr>
<td>Properties in VBS:FillStyle2</td>
<td>421</td>
</tr>
<tr>
<td>Properties in VBS:FineGrid</td>
<td>422</td>
</tr>
<tr>
<td>Properties in VBS:FineGridValue</td>
<td>422</td>
</tr>
<tr>
<td>Properties in VBS:FineGridValueX</td>
<td>422</td>
</tr>
<tr>
<td>Properties in VBS:FineGridValueY</td>
<td>423</td>
</tr>
</tbody>
</table>
Properties in VBS:FineGridX, 423
Properties in VBS:FineGridY, 423
Properties in VBS:FlashBackColor, 423
Properties in VBS:FlashBorderColor, 424
Properties in VBS:FlashFlashPicture, 424
Properties in VBS:FlashForeColor, 424
Properties in VBS:FlashPicReferenced, 425
Properties in VBS:FlashPicTransColor, 425
Properties in VBS:FlashRate, 426
Properties in VBS:FlashRateBackColor, 427
Properties in VBS:FlashRateBorderColor, 427
Properties in VBS:FlashRateFlashPic, 428
Properties in VBS:FlashRateForeColor, 428
Properties in VBS:Flip, 429
Properties in VBS:FocusColor, 430
Properties in VBS:FocusRect, 430
Properties in VBS:FocusWidth, 430
Properties in VBS:Font, 431
Properties in VBS:FontBold, 432
Properties in VBS:FontItalic, 432
Properties in VBS:FontName, 433
Properties in VBS:FontPosition, 433
Properties in VBS:FontSize, 433
Properties in VBS:FontStrikeThru, 434
Properties in VBS:FontUnderline, 434
Properties in VBS:ForeColor, 435
Properties in VBS:ForeFlashColorOff, 435
Properties in VBS:ForeFlashColorOn, 436
Properties in VBS:FrameColor, 436
Properties in VBS:FrameColorDown, 436
Properties in VBS:FrameColorUp, 437
Properties in VBS:FramePicture, 437
Properties in VBS:FrameScale, 437
Properties in VBS:FrameWidth, 438
Properties in VBS:FreezeProviderConnections, 438
Properties in VBS:GraphDirection, 439
Properties in VBS:GridLineHorz, 440
Properties in VBS:Gridlines, 439, 440
Properties in VBS:GridLinesValueX, 440
Properties in VBS:GridLinesValueY, 441
Properties in VBS:GridlinesX, 441
Properties in VBS:GridLinesY, 441
Properties in VBS:GridLineValue, 442
Properties in VBS:GridLineVert, 442
Properties in VBS:HandFillColor, 442
Properties in VBS:Handtype, 443
Properties in VBS:HeaderSort, 443
Properties in VBS:Height, 443
Properties in VBS:HiddenInput, 444
Properties in VBS:Hotkey, 450
Properties in VBS:HourNeedleHeight, 450
Properties in VBS:HourNeedleWidth, 450
Properties in VBS:Hysteresis, 451
Properties in VBS:HysteresisRange, 451
Properties in VBS:InnerBevelOffset, 453
Properties in VBS:InnerBevelStyle, 453
Properties in VBS:InnerBevelWidth, 453
Properties in VBS:InsertData, 454
Properties in VBS:ItemBorderColor, 455
Properties in VBS:ItemBorderBackColor, 455
Properties in VBS:ItemBorderStyle, 455
Properties in VBS:ItemBorderWidth, 456
Properties in VBS:ItemVisible, 456
Properties in VBS:Label, 457
Properties in VBS:LabelColor, 457
Properties in VBS:LabelX, 457
Properties in VBS:LabelY, 458
Properties in VBS:LanguageSwitch, 458
Properties in VBS:Language-IDs, 458
Properties in VBS:LastError, 459
Properties in VBS:Layer, 460
Properties in VBS:Layer00Checked, 461
Properties in VBS:Layer00Color, 465
Properties in VBS:Layer00Value, 472
Properties in VBS:Layer01Checked, 461
Properties in VBS:Layer01Color, 465
Properties in VBS:Layer01Value, 473
Properties in VBS:Layer02Checked, 462
Properties in VBS:Layer02Color, 465
Properties in VBS:Layer02Value, 473
Properties in VBS:Layer03Checked, 463
Properties in VBS:Layer03Color, 466
Properties in VBS:Layer03Value, 473
Properties in VBS:Layer04Checked, 464
Properties in VBS:Layer04Color, 467
Properties in VBS:Layer04Value, 474
Properties in VBS:Layer05Checked, 465
Properties in VBS:Layer05Color, 466
Properties in VBS:Layer05Value, 474
Properties in VBS:Layer06Checked, 463
Properties in VBS:Layer06Color, 467
Properties in VBS:Layer06Value, 474
Properties in VBS:Layer07Checked, 463
Properties in VBS:Layer07Color, 467
Properties in VBS:Layer07Value, 474
Properties in VBS:Layer08Checked, 464
Properties in VBS:Layer08Color, 467
Properties in VBS:Layer08Value, 475
Properties in VBS:Layer09Checked, 465
Properties in VBS:Layer09Color, 468
Properties in VBS:Layer09Value, 475
Properties in VBS:Layer10Checked, 464
Properties in VBS:PictureSelected, 538
Properties in VBS:PictureThumb, 538
Properties in VBS:PictureUnselected, 539
Properties in VBS:PictureUp, 539
Properties in VBS:PicUpReferenced, 539
Properties in VBS:PicUpTransparent, 540
Properties in VBS:PicUpUseTransColor, 540
Properties in VBS:PointCount, 541
Properties in VBS:Position, 541
Properties in VBS:Precisions, 542
Properties in VBS:PrecisionX, 542
Properties in VBS:PrecisionY, 542
Properties in VBS:PredefinedAngles, 543
Properties in VBS:Pressed, 543
Properties in VBS:PrintJob, 544
Properties in VBS:Process, 544
Properties in VBS:ProjectPath, 545
Properties in VBS:ProviderClsID, 545
Properties in VBS:ProviderType, 545
Properties in VBS:QualityCode, 546
Properties in VBS:Radius, 547
Properties in VBS:RadiusHeight, 547
Properties in VBS:RadiusWidth, 548
Properties in VBS:RangeMax, 548
Properties in VBS:RangeMin, 548
Properties in VBS:Rectangular, 549
Properties in VBS:ReferenceRotationLeft, 549
Properties in VBS:ReferenceRotationTop, 549
Properties in VBS:RelayCurves, 550
Properties in VBS:Relevant, 550
Properties in VBS:Replacement, 550
Properties in VBS:ReplacementColor, 551
Properties in VBS:RightComma, 551
Properties in VBS:RoundCornerHeight, 553
Properties in VBS:RoundCornerWidth, 553
Properties in VBS:RulerPrecisions, 556
Properties in VBS:RulerPrecisionX, 556
Properties in VBS:RulerPrecisionY, 556
Properties in VBS:SameSize, 557
Properties in VBS:ScaleColor, 558
Properties in VBS:ScaleTicks, 558
Properties in VBS:Scaling, 558
Properties in VBS:Scaling, 558
Properties in VBS:ScalingType, 558
Properties in VBS:ScalingTypeX, 559
Properties in VBS:ScalingTypeY, 560
Properties in VBS:Screen, 560
Properties in VBS:ScreenItems, 561
Properties in VBS:ScreenName, 560
Properties in VBS:Screen, 309
Properties in VBS:ScrollBars, 561
Properties in VBS:SecondNeedleHeight, 563
Properties in VBS:SecondNeedleWidth, 563
Properties in VBS:SelBGColor, 563
Properties in VBS:SelectionMode, 566
Properties in VBS:SelectionRectColor, 567
Properties in VBS:SelectionRectWidth, 567
Properties in VBS:SelectionType, 568
Properties in VBS:SelTextColor, 569
Properties in VBS:ServerData, 569
Properties in VBS:ServerNames, 571
Properties in VBS:ServerPrefix, 571
Properties in VBS:ShowBar, 572
Properties in VBS:ShowDanger, 573
Properties in VBS:ShowDecimalPoint, 573
Properties in VBS:ShowNormal, 573
Properties in VBS:ShowPeak, 574
Properties in VBS:ShowPosition, 574
Properties in VBS:ShowRulerImmediately, 575
Properties in VBS:ShowThumb, 577
Properties in VBS:ShowWarning, 579
Properties in VBS:SignificantMask, 579
Properties in VBS:SmallChange, 580
Properties in VBS:SourceBeginTime, 581
Properties in VBS:SourceEndTime, 582
Properties in VBS:SourceNumberOfUAValues, 582
Properties in VBS:SourceNumberOf UA Values, 583
Properties in VBS:SourceTagNameX, 583
Properties in VBS:SourceTagNameY, 584
Properties in VBS:SourceTagProviderDataX, 584
Properties in VBS:SourceTagProviderDataY, 584
Properties in VBS:SourceTimeRange, 585
Properties in VBS:SourceUAArchive, 585
Properties in VBS:SourceUAArchiveStartID, 585
Properties in VBS:SourceUAColumnX, 586
Properties in VBS:SourceUAColumnY, 586
Properties in VBS:SquareExtent, 587
Properties in VBS:StartAngle, 587
Properties in VBS:StatusBar, 588
Properties in VBS:StatusBarPanels, 592
Properties in VBS:StatusBarStretch, 594
Properties in VBS:Tag, 695
Properties in VBS:TagName, 596
Properties in VBS:TagPrefix, 596
Properties in VBS:TagName, 596
Properties in VBS:ProviderClsid, 597
Properties in VBS:Tags, 597
Properties in VBS:Template, 598
Properties in VBS:Text, 598
Properties in VBS:ThumbBackColor, 599
Properties in VBS:TicColor, 599
Properties in VBS:TicFont, 599
Properties in VBS:TicFont, 599
Properties in VBS:TicksColor, 601
Properties in VBS:TickStyle, 602
Properties in VBS:TicOffset, 600
Properties in VBS:TicTextColor, 600
Properties in VBS:TicTextOffset, 600
Properties in VBS:TicWidth, 601
Properties in VBS:TimeAxisFormat, 605
Properties in VBS:TimeAxisX, 609
Properties in VBS:TimeColumnAlignment, 611
Properties in VBS:TimeFormat, 619
Properties in VBS:TimeJump, 619
Properties in VBS:TimeOverlap, 620
Properties in VBS:TimeRange, 621
Properties in VBS:TimeRangeBase, 621
Properties in VBS:TimeRangeFactor, 622
Properties in VBS:TimeStamp, 622
Properties in VBS:TimeZone, 624
Properties in VBS:TitleCut, 625
Properties in VBS:Titleline, 626
Properties in VBS:Toggle, 627
Properties in VBS:ToleranceHigh, 628
Properties in VBS:ToleranceLow, 628
Properties in VBS:Toolbar, 628
Properties in VBS:ToolbarAlignment, 629
Properties in VBS:ToolbarButtons, 638
Properties in VBS:ToolbarHotKeys, 640
Properties in VBS:ToolTipText, 641
Properties in VBS:Top, 643
Properties in VBS:TopConnectedConnectionPointIndex, 644
Properties in VBS:TopConnectedObjectName, 644
Properties in VBS:Transparent, 644
Properties in VBS:Trend, 645
Properties in VBS:TrendColor, 647
Properties in VBS:Type, 666
Properties in VBS:TypeAlarmHigh, 668
Properties in VBS:TypeAlarmLow, 668
Properties in VBS:TypeLimitHigh4, 669
Properties in VBS:TypeLimitHigh5, 669
Properties in VBS:TypeLimitLow4, 669
Properties in VBS:TypeLimitLow5, 670
Properties in VBS:TypeToleranceHigh, 670
Properties in VBS:TypeToleranceLow, 670
Properties in VBS:TypeWarningHigh, 670
Properties in VBS:TypeWarningLow, 670
Properties in VBS:UnitColor, 671
Properties in VBS:UnitFont, 671
Properties in VBS:UnitOffset, 672
Properties in VBS:UnitText, 672
Properties in VBS:UnselBGColor, 672
Properties in VBS:UnselTextColor, 673
Properties in VBS:UpdateCycle, 673
Properties in VBS:UpperLimit, 673
Properties in VBS:UpperLimitColor, 674
Properties in VBS:UpperLimitValue, 674
Properties in VBS:UserValue1, 677
Properties in VBS:UserValue2, 677
Properties in VBS:UserValue3, 677
Properties in VBS:UserValue4, 678
Properties in VBS:Value, 680
Properties in VBS:ValueColumnAlignment, 688
Properties in VBS:ValueMax, 695
Properties in VBS:ValueMin, 695
Properties in VBS:Visible, 696
Properties in VBS:Warning, 697
Properties in VBS:WarningColor, 697
Properties in VBS:WarningHigh, 697
Properties in VBS:WarningLow, 698
Properties in VBS:Width, 698
Properties in VBS:WindowBorder, 699
Properties in VBS:WindowsStyle, 700
Properties in VBS:WindowType, 701
Properties in VBS:WithAxes, 701
Properties in VBS:WithLabels, 701
Properties in VBS:ZeroPoint, 709
Properties in VBS:ZeroPointValue, 709
Properties in VBS:Zoom, 709
Properties in VBS:FillingDirection, 2234
Property, 2464
Protect VBA Code, 1615
Push Button Control, 269
putc, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Index

Q
qsort, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
QuitHorn, 782
QuitSelected, 783
QuitVisible, 783

R
rand, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ReadTags, 787
realloc, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Rename:Module, 50
Renaming, 50, 76
Renaming:Action, 76
Renaming:Procedure, 50
ReportJob, 968
rewind, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
RPTJobPreview, 1000
RPTJobPrint, 1000
RptShowError, 1001
RulerBlock object, 226
RulerColumn object, 227
RulerControl, 272
Runtime, 78
Activating Global Actions, 78
S
Save, 63
  Module, 48
  procedure, 48
Save As...
  Applying, 886
Save:Action, 63
Saving file as, 922
Screen Object, 131
ScreenItem, 256, 262, 272, 280, 295
  Control, 181
  Controls, 218
  Customized object, 298
  Faceplate instance, 187
  WinCC Alarm Control, 286
  WinCC AlarmControl, 241
ScreenItem Object, 125
ScreenItem:3D bar, 169
ScreenItem:Application window, 173
ScreenItem:Bar, 174
ScreenItem:Button, 201
ScreenItem:Check Box, 204
ScreenItem:Circle, 149
ScreenItem:Circle segment, 152
ScreenItem:Circular arc, 151
ScreenItem:Combox, 189
ScreenItem:Connector, 167
ScreenItem:Double T-piece, 216
ScreenItem:Ellipse, 144
ScreenItem:Ellipse arc, 146
ScreenItem:Ellipse segment, 147
ScreenItem:Graphic Object, 187
ScreenItem:Group, 300
ScreenItem:Group Display, 194
ScreenItem:HMI Symbol Library, 239
ScreenItem:I/O Field, 184
ScreenItem:Line, 154
ScreenItem:ListBox, 190
ScreenItem:Multiline Text, 191
ScreenItem:Object types, 143
ScreenItem:OLE Element, 191
ScreenItem:Picture window, 179
ScreenItem:Polygon, 156
ScreenItem:Polygon Tube, 215
ScreenItem:Polyline, 158
ScreenItem:Radio Box, 207
ScreenItem:Rectangle, 159
ScreenItem:Roundbutton, 209
ScreenItem:Rounded Rectangle, 162
ScreenItem:Slider, 212
ScreenItem:Static Text, 165
ScreenItem:Status Display, 199
ScreenItem:Text list, 196
ScreenItem:T-piece, 216
ScreenItem:Tube Arc, 217
ScreenItem:WinCC Digital Analog Clock, 246
ScreenItem:WinCC Function Trend Control, 288
ScreenItem:WinCC FunctionTrendControl, 248
ScreenItem:WinCC Gauge Control, 254
ScreenItem:WinCC Online Table Control, 292
ScreenItem:WinCC OnlineTableControl, 257
ScreenItem:WinCC Push Button Control, 269
ScreenItem:WinCC Slider Control, 277
ScreenItems object (list), 129
Screns Object (List), 134
Script file, 92
Script File:Action Names, 94
Script file:open in Debugger, 96
Script File:Permissible action name length, 94
Script File:Set bookmark, 100
Script files of VBScripts, 92
SelectedStatisticArea, 799
ServerExport, 799
ServerImport, 799
Set_Focus, 1287
SetActualPointLeft, 1295
SetActualPointTop, 1296
SetAlarmHigh, 1319
SetAlarmLow, 1320
SetAlignment, 1247
SetAlignmentLeft, 1288
SetAlignmentTop, 1289
SetAssumeOnExit, 1312
SetAssumeOnFull, 1312
SetAverage, 1355
SetAxisSection, 1248
SetBackBorderWidth, 1396
SetBackColor, 1257
SetBackColor2, 1258
SetBackColor3, 1258
SetBackColorBottom, 1259
SetBackColorTop, 1260
SetBackFlashColorOff, 1277
SetBackFlashColorOn, 1278
SetBasePicTransColor, 1390
SetBasePicUseTransColor, 1391
SetBitNumber, 1313
SetBorderBackColor, 1261
SetBorderBackColor, 1261
SetBorderColor, 1262
SetBorderColorBottom, 1262
SetBorderColorTop, 1263
SetBorderEndStyle, 1397
Index

SetBorderFlashColorOff, 1279
SetBorderFlashColorOn, 1279
SetBorderStyle, 1398
SetBoxWidth, 1398
SetBoxAlignment, 1399
SetBoxCount, 1297
SetBoxType, 1355
SetBuf, 1011, 1012, 1013, 1014, 1015, 1016, 1017,
1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025,
1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041,
1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057,
1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065,
1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073,
1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095
SetButtonColor, 1264
SetCheckAlarmHigh, 1321
SetCheckAlarmLow, 1321
SetCheckLimitHigh4, 1322
SetCheckLimitHigh5, 1323
SetCheckLimitLow4, 1324
SetCheckLimitLow5, 1324
SetCheckToleranceHigh, 1325
SetCheckToleranceLow, 1326
SetCheckWarningHigh, 1327
SetCheckWarningLow, 1327
SetClearOnError, 1314
SetClearOnNew, 1315
SetColorAlarmHigh, 1328
SetColorAlarmLow, 1329
SetColorBottom, 1265
SetColorChangeType, 1356
SetColorLimitHigh4, 1330
SetColorLimitHigh5, 1331
SetColorLimitLow4, 1331
SetColorLimitLow5, 1332
SetColorToleranceHigh, 1333
SetColorToleranceLow, 1334
SetColorTop, 1266
SetColorWarningHigh, 1335
SetColorWarningLow, 1335
SetColorCursorControl, 1357
SetColorMode, 1358
SetDirection, 1298
SetEditAtOnce, 1359
SetEndAngle, 1299
SetExponent, 1249
SetExtendedOperation, 1359
SetFillColor, 1266
SetFilling, 1275
SetFillingIndex, 1276
SetFillStyle, 1400
SetFillStyle2, 1401
SetFlashBackColor, 1280
SetFlashBorderColor, 1281
SetFlashFlashPicture, 1392
SetFlashForeColor, 1282
SetFlashPicTransColor, 1392
SetFlashPicUseTransColor, 1393
SetFlashRateBackColor, 1282
SetFlashRateBorderColor, 1283
SetFlashRateFlashPic, 1394
SetFontBold, 1290
SetFontItallic, 1290
SetFontName, 1291
SetFontSize, 1292
SetFontUnderline, 1293
SetForeColor, 1267
SetForeFlashColorOff, 1285
SetForeFlashColorOn, 1286
SetHeight, 1299
SetHiddenInput, 1315
SetHysteresis, 1360
SetHysteresisRange, 1361
SetIndex, 1395
SetItemBorderBackColor, 1268
SetItemBorderColor, 1269
SetItemBorderStyle, 1402
SetItemBorderWidth, 1402
SetLanguage, 1513
SetLeft, 1300
SetLeftComma, 1250
SetLimitHigh4, 1336
SetLimitHigh5, 1337
SetLimitLow4, 1338
SetLimitLow5, 1338
SetLimitMax, 1339
SetLimitMin, 1340
SetLink, 1353
SetLongStrokesBold, 1251
SetLongStrokesOnly, 1252
SetLongStrokesSize, 1252
SetMarker, 1341
SetMax, 1362
SetMin, 1362
SetNumberLines, 1316
SetOffsetLeft, 1363
SetOffsetTop, 1364
SetOperation, 1364
SetOperationMessage, 1365
SetTypeLimitHigh4, 1345
SetTypeLimitHigh5, 1346
SetTypeLimitLow4, 1346
SetTypeLimitLow5, 1347
SetTypeToleranceHigh, 1348
SetTypeToleranceLow, 1349
SetTypeWarningHigh, 1350
SetTypeWarningLow, 1351
SetUnselBGColor, 1273
SetUnselTextColor, 1274
setvbuf, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
SetVisible, 1372
SetWarningHigh, 1352
SetWarningLow, 1352
SetWidth, 1310
SetWindowsStyle, 1405
SetZeroPoint, 1311
SetZeroPointValue, 1373
SetZoom, 1373
ShowBadTagState, 2418
ShowColumnSelection, 800
ShowComment, 800
ShowDisplayOptionsDialog, 801
ShowEmergencyQuitDialog, 801
ShowHelp, 801
ShowHideList, 802
ShowHitList, 802
ShowInfoText, 803
ShowLockDialog, 803
ShowLockList, 804
ShowLongTermArchiveList, 804
ShowMessageList, 804
ShowPercentageAxis, 805
ShowPropertyDialog, 805
ShowSelectArchive, 806
ShowSelection, 806
ShowSelectionDialog, 807
ShowSelectTimeBase, 806
ShowShortTermArchiveList, 807
ShowSort, 808
ShowSortDialog, 808
ShowTagSelection, 808
ShowTimebaseDialog, 809
ShowTimeSelection, 809
ShowTrendSelection, 810
Simulation, 2419
SimulationBit, 2419
sin, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
sinh, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Slider control, 277
Smart object, 1662
Editing with VBA, 1664
Smart Objects
Combobox,
Control, 181
Faceplate instance,
Smart objects: 3D bar, 169
Smart objects: Application window, 173
Smart objects: Bar, 174
Smart Objects: Graphic Object, 187
Smart objects: Group Display, 194
Smart objects: I/O Field, 184
Smart objects: List Box, 190
Smart Objects: Multiline Text, 191
Smart Objects: Picture window, 179
Smart objects: Status Display, 199
Smart objects: Text list, 196
Smart tag object, 136
SmartTag property, 580

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA
T

Table Control, 292

Tag, 24, 64, 72, 2435, 2562
  Change limits with VBA, 2562
  Change type with VBA, 2562
  Configure a Trigger to an Action, 72
  Creating with VBA, 2562
  Define type with VBA, 2562
  Deleting with VBA, 2562
  Read limits with VBA, 2562
  Read properties with VBA, 2562
  Tag Logging, 2573
    Create archive tag with VBA, 2573
    Create process value archive with VBA, 2573
    Delete archive tag with VBA, 2573
    Delete process value archive with VBA, 2573
    Edit archive tag with VBA, 2573
    Edit process value archive with VBA, 2573
  Tag Object, 137
  Tag:global in VBS, 24
  tagname, 2436
  Tags Object (List), 140
  TagScaleParam1 properties, 2437
  TagScaleParam2 properties, 2437
  TagScaleParam3 properties, 2438
  TagScaleParam4 properties, 2438
  TagSet Object (List), 141
  tagtype, 2438
  tan, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
  Symbol Library, 239
    Access with VBA,
    Copying an object with VBA,
    Creating a folder with VBA,
    Deleting a folder with VBA,
    Editing with VBA,
    Paste Object into a Picture with VBA,
  Syntax, 63
  Syntax:Control, 63
  Syntax:Error, 63
  SysFree, 1011
  SysMalloc, 1011
  system, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
  System behavior, 905

Text library, 2608
  Changing text with VBA, 2608
  Creating language with VBA, 2608
Creating text with VBA, 2608
Deleting text with VBA, 2608
Read text with VBA, 2608
Read TextID with VBA, 2608

TextBiblIDs, 2439

time, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095

TimeAxis object, 230
TimeColumn object, 231
Timer, 64, 70
    Configure a Trigger to an Action, 70
TitleBackColorActiveEnd property, 2440
TitleBackColorActiveStart property, 2440
TitleBackColorInactiveEnd property, 2440
TitleBackColorInactiveStart property, 2440
TitleForeColorActive property, 2441
TitleForeColorInactive property, 2441
TlgGetColumnPosition, 998
TlgGetNumberOfColumns, 992
TlgGetNumberOfRows, 993
TlgGetNumberOfTrends, 993
TlgGetRowPosition, 994
TlgGetRulerArchivNameTrend, 994
TlgGetRulerTimeTrend, 995
TlgGetRulerValueTrend, 996
TlgGetRulerVariableNameTrend, 997
TlgGetTextAtPos, 997
TlgGetXMLAtPos, 997
TlgTableWindowPressEditRecordButton, 969
TlgTableWindowPressFirstButton, 970
TlgTableWindowPressInsertRecordButton, 971
TlgTableWindowPressLastButton, 972
TlgTableWindowPressNextButton, 972
TlgTableWindowPressNextItemButton, 973
TlgTableWindowPressOpenArchiveVariableSelectio
dDlgButton, 974
TlgTableWindowPressOpenDlgButton, 974
TlgTableWindowPressOpenItemSelectDlgButton, 975
TlgTableWindowPressOpenTimeSelectDlgButton, 976
TlgTableWindowPressPrevButton, 976
TlgTableWindowPressPrevItemButton, 977
TlgTableWindowPressRemoveRecordButton, 978
TlgTableWindowPressStartStopButton, 978
TlgTrendWindowActivateCurve, 999
TlgTrendWindowPressFirstButton, 979
TlgTrendWindowPressHelpButton, 979
TlgTrendWindowPressLastButton, 980
TlgTrendWindowPressLinealButton, 981
TlgTrendWindowPressNextButton, 981
TlgTrendWindowPressNextItemButton, 982
TlgTrendWindowPressOneToOneButton, 983
TlgTrendWindowPressOpenArchiveVariableSelectio
dDlgButton, 983
TlgTrendWindowPressOpenDlgButton, 984
TlgTrendWindowPressOpenItemSelectDlgButton, 985
TlgTrendWindowPressOpenTimeSelectDlgButton, 985
TlgTrendWindowPressPrevButton, 986
TlgTrendWindowPressPrevItemButton, 987
TlgTrendWindowPressPrintButton, 987
TlgTrendWindowPressStartStopButton, 989
TlgTrendWindowPressStatsResultButton, 989
TlgTrendWindowPressStatsSelectRangeButton, 990
TlgTrendWindowPressZoomInButton, 991
TlgTrendWindowPressZoomOutButton, 991
tmpfile, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
TlgTrendWindowPressStartStopButton, 978
TlgTrendWindowPressStatsResultButton, 989
TlgTrendWindowPressStatsSelectRangeButton, 990
TlgTrendWindowPressZoomInButton, 991
TlgTrendWindowPressZoomOutButton, 991
tmpnam, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
TlgTrendWindowPressStartStopButton, 978
TlgTrendWindowPressStatsResultButton, 989
TlgTrendWindowPressStatsSelectRangeButton, 990
TlgTrendWindowPressZoomInButton, 991
TlgTrendWindowPressZoomOutButton, 991

WinCC: Scripting (VBS, ANSI-C, VBA)
System Manual, 02/2016, A5E37536061-AA

2681
Toolbar, 1627
  Add icon, 1637
  application-specific, 1627
  Assigning help text,
  Assigning status text,
  Assigning VBA macro,
  Configuring,
  Insert,
  picture-specific, 1627
  Placement, 1627
  Properties, 1627
  user-defined, 1627
  Working with toolbars, 882
Toolbar:GSC Diagnostics, 83
ToolBarButton object, 232
ToolBarItemType, 2444
ToLower, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
Toolbar:GSC Diagnostics, 83
ToolBarButton object, 232
ToolBarItemType, 2444
ToLower, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
ToolBar:GSC Diagnostics, 83
ToolBarButton object, 232
ToolBarItemType, 2444
ToLower, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
TraceText, 1515
TraceTime, 1516
TransformDisplayCoordinate, 1892
TransformPixelCoordinate, 1892
trend, 2448
Trend object, 233
TrendWindow object, 234
Trigger, 64
  Changing, 920
  Deleting, 921
  Effect on actions, 914
  Picture cycle, 1708
  Standard cycle, 1708
Tag, 1708
  Window Cycle, 1708
Trigger type, 864, 914
Triggers, 70, 72, 1707
  Adding type "Timer", 70
  Adding type Tag, 72
  Changing, 74
  Configuring with VBA, 1707
  Deleting, 75
Tube objects:Double T-piece, 216
Tube objects:Polygon tube, 215
Tube objects:T-piece, 216
Tube objects:Tube Arc, 217
TubeArcObject object, 2088
TubeDoubleTeeObject object, 2090
TubePolyline object, 2092
TubeTeeObject object, 2094
U
ungetc, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095
UnhideAlarm, 812
UnlockAlarm, 812
UsedLanguage, 2464
UseEventState, 2464
UseGlobalAlarmClasses, 2465
UseGlobalSettings, 2466
User interface language, 1624
  Access with VBA, 1624
UserArchiveControl, 280
User-defined menu, 1619, 1627
  Add menu entry, 1631
  Assigning help text, 1639
  Assigning status text, 1639
  Assigning VBA macro, 1642
  Configuring, 1626
  Creating, 1629
  creating multiple languages, 1633
  Placement, 1627
  Properties, 1627
User-defined toolbar, 1619, 1627
  Add icon, 1637
Assigning help text, 1639
Assigning status text, 1639
Assigning VBA macro, 1642
Configuring, 1626
Creating, 1635
Placement, 1627
Properties, 1627
UseValueText, 2469

V
Validity range, 874
ValueAxis object, 235
ValueColumn object, 236
VB action, 1705
   Configuring on an event with VBA, 1705
VB Scripts, 1697
   Dynamize property with VBA, 1697
VBA, 1614
   Access to a copy of the pictures, 1654
   Access to external applications, 1713
   Access to group objects, 1674
   Access to layers, 1654
   Access to objects in the Graphics Designer, 1660
   Access to other programs, 1713
   Access to the component library, 1646
   Access to the symbol library, 1646
   Accessing MS Excel with VBA, 1714
   Action configuring, 1699
   Add menu entry to user-defined menu, 1631
   Adding a New Icon to the used-defined
   Toolbar, 1637
   Alarm Logging, 2622
   Assign help text to user-defined menu, 1639
   Assign help text to user-defined toolbar, 1639
   Assigning VBA macro to user-defined
   menu, 1642
   Assigning VBA macro to user-defined
   toolbar, 1642
   C Scripts (Delimitation), 1614
   Configure user-defined menu, 1626
   Configure user-defined toolbar, 1626
   Configuring a Direct Connection, 1701
   Configuring C Action on an Event, 1703
   Configuring Dynamics in a Property with C
   Script, 1694
   Configuring Dynamics in a Property with Dynamic
   Dialog, 1691
   Configuring Dynamics in a Property with VB
   Script, 1697
   Configuring event-controlled actions, 1699
   Configuring for multiple languages, 1624
Configuring triggers, 1707
Configuring VB Action on an Event, 1705
Controlling the visibility of layers, 1657
Create an application-specific menu, 1657
Create application-specific toolbar, 1635
Create user-defined menu, 1629
Create user-defined menu in multiple
languages, 1633
Create user-defined toolbar, 1635
Creating a group object, 1676
Customized Object, 1682
Deleting Group Objects, 1676
Dynamic Wizards (Delimitation), 1614
Dynamization, 1686
Edit component library, 1649
Edit layers, 1657
Edit Smart object, 1664
Edit standard object, 1664
Edit Windows object, 1664
Editing a copy of a picture, 1658
Editing a group object, 1676
Editing Customized Objects, 1683
Editing objects in a group object, 1679
Editing objects in Graphic Designer, 1660
Editing objects in Graphics Designer, 1660
Editing picture, 1654
Editing symbol library, 1649
Event Handling, 1709
Executing VBA macros, 1619
Export code, 1618
global VBA code, 1615
Group Objects, 1674
HMIGO class, 2560
Import code, 1618
in Graphics Designer, 1621
in other WinCC Editors, 2560
Inserting an ActiveX control into a picture, 1671
Language-dependent configuration, 1624
Making properties dynamic, 1687
ODK (Delimitation), 1614
OLE Element, 1669
Paste Object from component library into a
Picture, 1652
Paste Object from symbol library into a
Picture, 1652
Paste Object into Picture, 1662
picture-specific VBA code, 1615
Project language, 1624
project-specific VBA Code, 1615
Tag Logging, 2573
Tag Management, 2562
Text library, 2608
WinCC Function Trend Control
StatusbarElement object, 229
ToolbarButton object, 232
Trend object, 233
TrendWindow object, 234
XAxis object, 237
YAxis object, 238
WinCC Online Table Control
Example of C script, 1579
StatusbarElement object, 229
TimeColumn object, 231
ToolbarButton object, 232
ValueColumn object, 236
VBS example, 851
WinCC Online Trend Control
Example of C script, 1581
StatusbarElement object, 229
ToolbarButton object, 232
Trend object, 233
TrendWindow object, 234
VBS example, 846
VBS example to setpoint trend, 849
WinCC Ruler Control
RulerBlock object, 226
RulerColumn object, 227
StatisticAreaColumn object, 228
StatisticResultColumn object, 229
StatusbarElement object, 229
ToolbarButton object, 232
WinCC User Archive Control
Column object, 221
StatusbarElement object, 229
ToolbarButton object, 232
WinCC: Graphic object types, 143
WinCC: WinCC Digital Analog Clock, 246
WinCC: WinCC Function Trend Control, 288
WinCC: WinCC Function Trend Control, 248
WinCC: WinCC Gauge Control, 254
WinCC: WinCC Online Table Control Online, 292
WinCC: WinCC Online Trend Control, 295
WinCC: WinCC Online Table Control, 257
WinCC: WinCC Online Trend Control, 262
WinCC: WinCC Push Button Control Controls: WinCC
Push Button Control, 269
WinCC: WinCC Ruler Control, 272
WinCC: WinCC User Archive Control, 280
Window, 81, 84
Windows object, 1662
  Editing with VBA, 1664
Windows Objects: Button, 201
Windows Objects: Check Box, 204
Windows Objects: Radio Box, 207

Windows Objects: Round Button, 209
Windows Objects: Slider, 212
WPFCtrl, 2104
WriteTag, 817
Writing tag values, 828

X
XAxis object, 237

Y
YAxis object, 238

Z
ZoomArea, 818
ZoomInOut, 818
ZoomInOutTime, 818
ZoomInOutValues, 819
ZoomInOutX, 819
ZoomInOutY, 820
ZoomMove, 820