SIEMENS

VBS for Creating Procedures and Actions 1

ANSI-C for Creating Functions and Actions 2

ANSI-C function descriptions 3

VBA for Automated Configuration 4

SIMATIC HMI

WinCC V7.0 SP1
MDM - WinCC: Scripting (VBS, ANSI-C, VBA)

System Manual

Print of the Online Help

11/2008

Legal information

Warning notice system

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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 for the specific task, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 08/2009

Copyright © Siemens AG 2008.
Technical data subject to change

Table of contents

1 VBS for Creating Procedures and Actions....13

1 Resources....13
1.1 VBS for Creating Procedures and Actions....13
1.2 Using Visual Basic Script in WinCC....14
1.3 Modules and Procedures....18
1.4 Actions....21
1.5 Multiple Use of Procedures and Actions....23
1.6 Use of CrossReference....25
1.7 Using Global Tags in VBS....27
1.8 VBScript Editors....28
1.8.1 VBScript Editors....28
1.8.2 Global Script Editor....30
1.8.3 Working in an Editing Window....32
1.8.4 Working with the Toolbars....36
1.8.5 Deleting Actions or Procedures....38
1.9 Creating and Editing Procedures....39
1.9.1 Creating and Editing Procedures....39
1.9.2 Creating a New Procedure....42
1.9.3 How to Write Procedure Codes....45
1.9.4 Using Standard and Project Procedures....47
1.9.5 Adding Module-related Information....49
1.9.6 Protecting a Module with a Password....51
1.9.7 Saving a Procedure....52
1.9.8 Renaming a Procedure or Module....54
1.10 Creating and Editing Actions....55
1.10.1 Creating and Editing Actions....55
1.10.2 Creating a New Action....59
1.10.3 How to Edit Actions....61
1.10.4 How to add information to the action....64
1.10.5 Protecting an Action with a Password....66
1.10.6 Saving Actions....67
1.10.7 Triggers....69
1.10.7.1 Triggers....69
1.10.7.2 Animation trigger....73
1.10.7.3 How to add a trigger of the type "Timer"....74
1.10.7.4 How to add a trigger of the type "Tag"....76
1.10.7.5 How to change a trigger....78
1.10.7.6 How to delete a trigger....79
1.10.8 How to Rename an Action....81
1.11 Activating Global Actions in Runtime....82

1.12 Diagnostics....83
1.12.1 Diagnostics....83
1.12.2 GSC Diagnostics....84
1.12.2.1 GSC Diagnostics....84
1.12.2.2 Inserting the GSC Diagnostics Window into a Picture....85
1.12.2.3 GSC Diagnostics Attributes....86
1.12.2.4 GSC Diagnostics Toolbar....86
1.12.3 GSC Runtime....87
1.12.3.1 GSC Runtime....87
1.12.3.2 Inserting the GSC Runtime Window into a Picture....88
1.12.3.3 GSC Runtime Attributes....89
1.12.4 Testing with the Debugger....90
1.12.4.1 Testing with the Debugger....90
1.12.4.2 How to Activate the Debugger....91
1.12.4.3 Principles of Debugging....93
1.12.4.4 Components of the Microsoft Script Debuggers....95
1.12.4.5 Structure of VBScript Files....97
1.12.4.6 Action and Procedure Names in the Debugger....99
1.12.4.7 Selecting a Script for Editing....100
1.12.4.8 Processing Scripts Step-by-Step....102
1.12.4.9 Setting Breakpoints....103
1.12.4.10 Deleting Breakpoints....104
1.12.4.11 How to Set Bookmarks in Scripts....105
1.12.4.12 How to Determine and Modify Tag and Property Values....106
1.12.4.13 Executing Script Commands....107
1.13 Printing VBScripts....108
1.14 VBS Reference....109
1.14.1 VBS Reference....109
1.14.2 Objects and Lists....113
1.14.2.1 Objects and Lists....113
1.14.2.2 Alarm object....115
1.14.2.3 Alarms object (list)....116
1.14.2.4 AlarmLogs Object....118
1.14.2.5 DataItem Object....119
1.14.2.6 DataLogs Object....121
1.14.2.7 DataSet-Objekt (List)....122
1.14.2.8 HMIRuntime Object....125
1.14.2.9 Item Object....126
1.14.2.10 Layer Object....127
1.14.2.11 Layers Object (Listing)....129
1.14.2.12 Logging Object....130
1.14.2.13 ProcessValue Object....131
1.14.2.14 ProcessValues Object (List)....132
1.14.2.15 Project Object....133
1.14.2.16 ScreenItem Object....134
1.14.2.17 ScreenItems Object (List)....138
1.14.2.18 Screen Object....140
1.14.2.19 Screens Object (List)....143
1.14.2.20 SmartTags Object....145
1.14.2.21 Tag Object....146
1.14.2.22 Tags Object (List)....149

1.14.2.23 TagSet Object (List)....151
1.14.3 Object Types of the ScreenItem Object....152
1.14.3.1 Object Types of the ScreenItem Object....152
1.14.3.2 Standard objects....154
1.14.3.3 Smart objects....182
1.14.3.4 Windows objects....214
1.14.3.5 Tube objects....226
1.14.3.6 Controls....230
1.14.3.7 Customized object....273
1.14.3.8 Group....274
1.14.4 Properties....276
1.14.4.1 Properties....276
1.14.4.2 A....276
1.14.4.3 B....295
1.14.4.4 C....320
1.14.4.5 D....350
1.14.4.6 E....359
1.14.4.7 F....368
1.14.4.8 G....386
1.14.4.9 H....390
1.14.4.10 I....399
1.14.4.11 L....405
1.14.4.12 M....435
1.14.4.13 N....452
1.14.4.14 O....455
1.14.4.15 P....470
1.14.4.16 Q....487
1.14.4.17 R....488
1.14.4.18 S....497
1.14.4.19 T....530
1.14.4.20 U....601
1.14.4.21 V....610
1.14.4.22 W....625
1.14.4.23 X - Z....630
1.14.5 Methods....638
1.14.5.1 Methods....638
1.14.5.2 Methods A to E....639
1.14.5.3 Get methods....648
1.14.5.4 Methods H to M....686
1.14.5.5 Methods N to R....694
1.14.5.6 Methods S to T....712
1.14.5.7 Methods U to Z....725
1.14.6 Appendix....732
1.14.6.1 Error Messages from Database Area....732

1.15 Examples of VBScript....734
1.15.1 Examples of VBScript....734
1.15.2 Examples in WinCC....734
1.15.2.1 Examples in WinCC....734
1.15.2.2 Example: Accessing objects in Graphics Designer....735
1.15.2.3 Example: Defining the color of objects....736
1.15.2.4 Example: How to Configure Language Changes....736
1.15.2.5 Example: Deactivating Runtime....737

1.15.2.6 Example: Configuring change picture globally....737
1.15.2.7 Example: Configuring Change Picture Via Property....738
1.15.2.8 Example: Configuring diagnostics output via Trace....738
1.15.2.9 Example: Writing tag values....739
1.15.2.10 Example: How to Read Tag Values....741
1.15.2.11 Example: Writing Object Properties....743
1.15.2.12 Example: How to Start an Action on the Server (Logging Object)....746
1.15.2.13 Dynamization of Controls....748
1.15.3 General Examples....755
1.15.3.1 General examples for VBScript....755
1.15.3.2 Example: Configuring a Database Connection with VBS....756
1.15.3.3 Example: Using the MS Automation interface....758
1.15.3.4 Example: Starting an external application....760

1.16 Basic Principles of VBScript....760
1.16.1 Basic Principles of VBScript....760
1.16.2 VBScript Basics....761

2 ANSI-C for Creating Functions and Actions....763

2 Resources....763

2.1 Creating Functions and Actions with ANSI-C....763

2.2 Creating Functions and Actions....763

2.3 Characteristics of Project Functions....766

2.4 Characteristics of Standard Functions....767

2.5 Characteristics of Internal Functions....768

2.6 Characteristics of Local Actions....769

2.7 Characteristics of Global Actions....770

2.8 How to Add Global Script Runtime to a Project's Startup List....771

2.9 Use of Global C-Tags....772

2.10 Use of DLLs in Functions and Actions....774

2.11 The Global Script Editor....775
2.11.1 The Global Script Editor....775
2.11.2 Working in the Edit Window....778
2.11.2.1 Working in the Edit Window....778
2.11.2.2 Editing Functions with the Keyboard....780
2.11.2.3 Editing Functions with the Mouse....780
2.11.3 Working with the Toolbars....781
2.11.4 How to Set Different Views....783
2.11.5 How to Set the Font Style....784
2.11.6 How to Use "Save As..."....784
2.11.7 How to Delete Actions or Project and Standard Functions....785
2.11.8 How to Generate a New Header....786
2.11.9 How to Compile All Functions....786
2.11.10 How to Search in Files....787
2.11.11 Printing Functions and Actions....788
2.11.11.1 Printing Functions and Actions....788
2.11.11.2 How to Set the Print Parameters....788

2.11.11.3 How to Open Page View....789
2.11.11.4 How to Print the Project Documentation....789

2.12 Creating and Editing Functions....790
2.12.1 Creating and Editing Functions....790
2.12.2 How to Create a New Function....793
2.12.3 How to Write Function Code....794
2.12.4 How to Use Internal Functions....795
2.12.5 How to Use Standard and Project Functions....796
2.12.6 Inserting Additional Function-Related Information....797
2.12.7 How to Protect a Function Against Unauthorized Access....798
2.12.8 How to Compile and Save a Function....799
2.12.9 How to Rename a Function....800
2.12.10 How to Use Functions from Other Sources....801

2.13 Creating and Editing Actions....802
2.13.1 How To Create and Edit Actions....802
2.13.2 WinCC Coding Rule....805
2.13.3 How to Create a New Action....806
2.13.4 How to Edit Actions....807
2.13.5 Inserting Additional Action-Related Information....807
2.13.6 How to Protect an Action Against Unauthorized Access....809
2.13.7 How to Compile and Save an Action....810
2.13.8 Triggers....811
2.13.8.1 Triggers....811
2.13.8.2 How to Add a New Trigger of the "Timer" Type....814
2.13.8.3 How to Add a New Trigger of the "Tag" Type....815
2.13.8.4 How to Change a Trigger....817
2.13.8.5 How to Delete a Trigger....818
2.13.9 How to Assign Authorizations....820
2.13.10 How to Export an Action....821
2.13.11 How to Import an Action....821
2.13.12 How to Rename an Action....822
2.13.13 How to Use Actions From Other Sources....823

2.14 Runtime Behavior of Actions....824
2.14.1 Runtime Behavior of Actions....824
2.14.2 GSC Runtime....825
2.14.2.1 GSC Runtime....825
2.14.2.2 How to Place GSC Runtime in a Process Picture....828
2.14.2.3 Attributes of GSC Runtime....829
2.14.2.4 How to Edit Actions....829
2.14.3 GSC Diagnose....830
2.14.3.1 GSC Diagnose....830
2.14.3.2 How to Place GSC Diagnose in a Process Picture?....831
2.14.3.3 Attributes of GSC Diagnose....832
2.14.3.4 The Toolbar of GSC Diagnose....832

2.15 ANSI-C function descriptions....833

3 ANSI-C function descriptions....835

3 Resources....835

3.1 lpszPictureName....835

3.2 Standard functions....835
3.2.1 Standard functions - short description....835
3.2.2 Alarm....836
3.2.2.1 AcknowledgeMessage....836
3.2.2.2 AXC_SetFilter....837
3.2.2.3 GCreateMyOperationMsg....838
3.2.2.4 GMsgFunction....840
3.2.3 Graphics....841
3.2.3.1 Graphics - short description....841
3.2.3.2 GetLinkedVariable....842
3.2.3.3 GetLocalPicture....842
3.2.3.4 GetParentPicture....843
3.2.3.5 GetParentPictureWindow....844
3.2.3.6 OpenPicture....845
3.2.3.7 Registry2....846
3.2.4 Report....847
3.2.4.1 Report - short description....847
3.2.4.2 RPTJobPreview....847
3.2.4.3 RPTJobPrint....848
3.2.4.4 RptShowError....849
3.2.5 WinCC....850
3.2.5.1 WinCC - short description....850
3.2.5.2 GetHWDiag....850
3.2.5.3 GetHWDiagLevel....851
3.2.5.4 GetKopFupAwl....853
3.2.5.5 GetKopFupAwlLevel....854
3.2.5.6 OnDeactivateExecute....855
3.2.5.7 OnErrorExecute....855
3.2.5.8 OnTime....857
3.2.6 Windows....857
3.2.6.1 Windows - short description....857
3.2.6.2 ProgramExecute....858
3.2.7 Obsolete functions....858
3.2.7.1 Alarm....858
3.2.7.2 Report....884
3.2.7.3 TagLog....885
3.3 Internal functions....919
3.3.1 Internal functions - short description....919
3.3.2 allocate....920
3.3.2.1 SysFree....920
3.3.2.2 SysMalloc....920
3.3.3 c_bib....921
3.3.3.1 c_bib - short description....921
3.3.3.2 ctype....921
3.3.3.3 math....930
3.3.3.4 memory....946
3.3.3.5 stdio....950
3.3.3.6 stdlib....971
3.3.3.7 string....986
3.3.3.8 time....997
3.3.4 graphics....1004

3.3.4.1 Graphics - short description....1004
3.3.4.2 get....1004
3.3.4.3 set....1168
3.3.5 tag....1354
3.3.5.1 tag - short description....1354
3.3.5.2 get....1355
3.3.5.3 set....1417
3.3.6 WinCC....1464
3.3.6.1 WinCC - short description....1464
3.3.6.2 system....1464
3.3.6.3 FillDiagnoseInTags....1468
3.3.6.4 GetServerTagPrefix....1469
3.3.6.5 TraceText....1469
3.3.6.6 TraceTime....1470

3.4 Appendix....1471
3.4.1 Examples....1471
3.4.1.1 Examles - A to G....1471
3.4.1.2 Examples - GetAlarmHigh to GetPropChar....1474
3.4.1.3 Examples - GetRangeMax to GetWidth....1489
3.4.1.4 Examples - H to S....1509
3.4.1.5 Examples - SetAlarmHigh to SetPropChar....1511
3.4.1.6 Examples - SetRangeMax to SetWidth....1521
3.4.1.7 Examples - T to Z....1530
3.4.2 Lists....1535
3.4.2.1 Bar direction....1535
3.4.2.2 Bar Scaling....1535
3.4.2.3 Flash frequencies....1536
3.4.2.4 I/O field, output format....1536
3.4.2.5 I/O field, data type of the field content....1537
3.4.2.6 I/O field, field type....1537
3.4.2.7 Element alignment in check boxes and radio boxes....1538
3.4.2.8 Color chart....1538
3.4.2.9 Format descriptors....1538
3.4.2.10 Fill pattern....1539
3.4.2.11 Line styles....1541
3.4.2.12 Line end style....1541
3.4.2.13 List types....1541
3.4.2.14 Language ID....1542
3.4.2.15 Text alignment....1543
3.4.2.16 Tag statuses....1543
3.4.3 Structure definitions....1544
3.4.3.1 Structure definition CCAPErrorExecute....1544
3.4.3.2 Structure definition CCAPTime....1545
3.4.3.3 Structure definition CMN_ERROR....1546
3.4.3.4 Structure definition DM_TYPEREF....1547
3.4.3.5 Structure definition DM_VAR_UPDATE_STRUCT....1548
3.4.3.6 Structure definition DM_VAR_UPDATE_STRUCTEX....1549
3.4.3.7 Structure definition DM_VARKEY....1550
3.4.3.8 Structure definition LINKINFO....1551
3.4.3.9 Structure definition MSG_FILTER_STRUCT....1552
3.4.3.10 Structure definition MSG_RTDATA_STRUCT....1556

4 VBA for Automated Configuration....1559

4 Resources....1559

4.1 Automated Configuring....1559

4.2 Introduction: Using VBA in WinCC....1560
4.2.1 Introduction: Using VBA in WinCC....1560
4.2.2 Differentiation: Using VBA....1561
4.2.3 Organizing VBA Code in a WinCC Project....1562
4.2.4 How to export and import VBA code....1565
4.2.5 Executing VBA Macros in Graphics Designer....1566

4.3 VBA in the Graphics Designer....1567
4.3.1 VBA in the Graphics Designer....1567
4.3.2 Adapting the Graphics Designer with VBA....1569
4.3.2.1 Adapting the Graphics Designer with VBA....1569
4.3.2.2 Language-Dependent Configuration with VBA....1570
4.3.2.3 Creating Customized Menus and Toolbars....1572
4.3.2.4 Accessing the component library with VBA....1593
4.3.3 Editing Pictures with VBA....1601
4.3.3.1 Editing Pictures with VBA....1601
4.3.3.2 How to Create Picture-Specific Menus and Toolbars....1602
4.3.3.3 Editing Layers with VBA....1603
4.3.3.4 Editing a Copy of a Picture with VBA....1605
4.3.4 Editing Objects with VBA....1606
4.3.4.1 Editing Objects with VBA....1606
4.3.4.2 Default objects, Smart objects, Windows objects and Tube objects....1609
4.3.4.3 Group Objects....1618
4.3.4.4 Customized Objects....1626
4.3.5 Creating Dynamics with VBA....1631
4.3.5.1 Creating Dynamics with VBA....1631
4.3.5.2 Configuring Dynamics in the Properties of Pictures and Objects....1632
4.3.5.3 Configuring Event-Driven Actions with VBA....1644
4.3.5.4 Editing Triggers....1652
4.3.6 Event Handling....1655
4.3.7 Accessing External Applications with VBA....1658
4.3.7.1 Accessing External Applications with VBA....1658
4.3.7.2 Example: Accessing MS Excel with VBA....1659

4.4 The object model of the Graphics Designer....1664
4.4.1 VBA Reference....1664
4.4.2 VBA Reference: ActionDynamic....1667
4.4.3 VBA Reference: HMIObjects....1669
4.4.4 VBA Reference: Languages....1671
4.4.5 Events....1671
4.4.5.1 A-D....1671
4.4.5.2 F-Z....1688
4.4.6 Methods....1704
4.4.6.1 A-C....1704
4.4.6.2 D-M....1744
4.4.6.3 O-Z....1780
4.4.7 Objects and Lists....1805
4.4.7.1 0-9, A-C....1805

4.4.7.2 D-I....1841
4.4.7.3 L-Q....1889
4.4.7.4 R-Z....1939
4.4.8 Properties....1995
4.4.8.1 A....1995
4.4.8.2 B....2016
4.4.8.3 C....2053
4.4.8.4 D....2089
4.4.8.5 E....2096
4.4.8.6 F....2104
4.4.8.7 G-H....2134
4.4.8.8 I - K....2144
4.4.8.9 L....2157
4.4.8.10 M....2206
4.4.8.11 N-O....2228
4.4.8.12 P-Q....2241
4.4.8.13 R....2268
4.4.8.14 S....2276
4.4.8.15 T....2299
4.4.8.16 U....2324
4.4.8.17 V....2331
4.4.8.18 W - Z....2406

4.5 AddIns....2414
4.5.1 AddIns....2414
4.5.2 Linking Add Ins....2415
4.5.3 Loading an Addin in the Graphics Designer....2417
4.5.4 Example: Creating Add Ins....2418
4.5.4.1 Example: Creating Add Ins....2418
4.5.4.2 Example: Creating an Add In with Visual Basic 6.0....2419

4.6 VBA in Other WinCC Editors....2424
4.6.1 VBA in Other WinCC Editors....2424
4.6.2 VBA in Tag Management....2426
4.6.2.1 VBA in Tag Management....2426
4.6.2.2 CloseTag Function....2429
4.6.2.3 CommitTag Function....2430
4.6.2.4 CreateTag Function....2432
4.6.2.5 DeleteTag Function....2434
4.6.2.6 GetTag Function....2435
4.6.2.7 ListTag function....2436
4.6.3 VBA im Tag Logging....2438
4.6.3.1 VBA in Tag Logging....2438
4.6.3.2 CloseTlgArchive-Funktion....2442
4.6.3.3 CloseTlgTag Function....2443
4.6.3.4 CommitTlgArchive Function....2445
4.6.3.5 CommitTlgTag Function....2446
4.6.3.6 CreateTlgArchive Function....2448
4.6.3.7 CreateTlgTag Function....2451
4.6.3.8 DeleteTlgArchive Function....2455
4.6.3.9 DeleteTlgTag Function....2457
4.6.3.10 GetTlgArchive Function....2458
4.6.3.11 GetTlgTag Function....2459

4.6.3.12 ListTlgArchive Function....2461
4.6.3.13 ListTlgTag Function....2463
4.6.4 VBA in the Text Library....2465
4.6.4.1 VBA in the Text Library....2465
4.6.4.2 CreateTextLanguage Function....2466
4.6.4.3 CreateText Function....2467
4.6.4.4 DeleteText Function....2468
4.6.4.5 DeleteTextLanguage Function....2470
4.6.4.6 GetText Function....2471
4.6.4.7 GetTextID Function....2473
4.6.4.8 ListText Function....2475
4.6.4.9 ModifyText Function....2476
4.6.5 VBA in Alarm Logging....2478
4.6.5.1 VBA in Alarm Logging....2478
4.6.5.2 CloseSingleAlarm Function....2481
4.6.5.3 CommitSingleAlarm Function....2482
4.6.5.4 CreateSingleAlarm Function....2483
4.6.5.5 DeleteSingleAlarm Function....2487
4.6.5.6 GetSingleAlarm Function....2488
4.6.5.7 ListSingleAlarm Function....2489
Index....2493

VBS for Creating Procedures and Actions 1

1 Resources

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.
- VBS 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.:

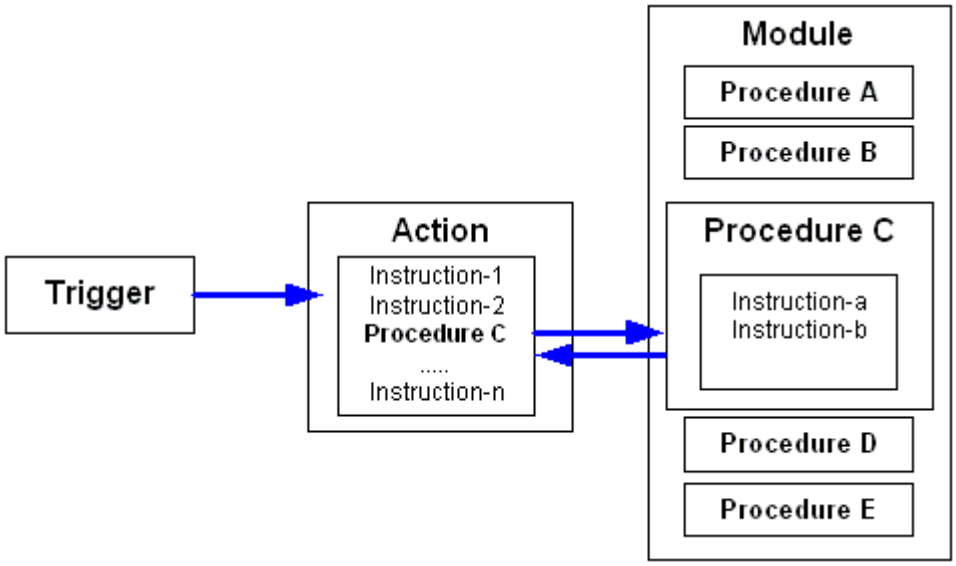
- VBS was developed for use in Internet, VBA for the automation of software applications.
- The data type of VBS tags is always VARIANT. VBA, on the other hand, differentiates the individual data types such as INT, DOUBLE, STRING, etc.
- Certain language constructs from VBA have been removed from or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

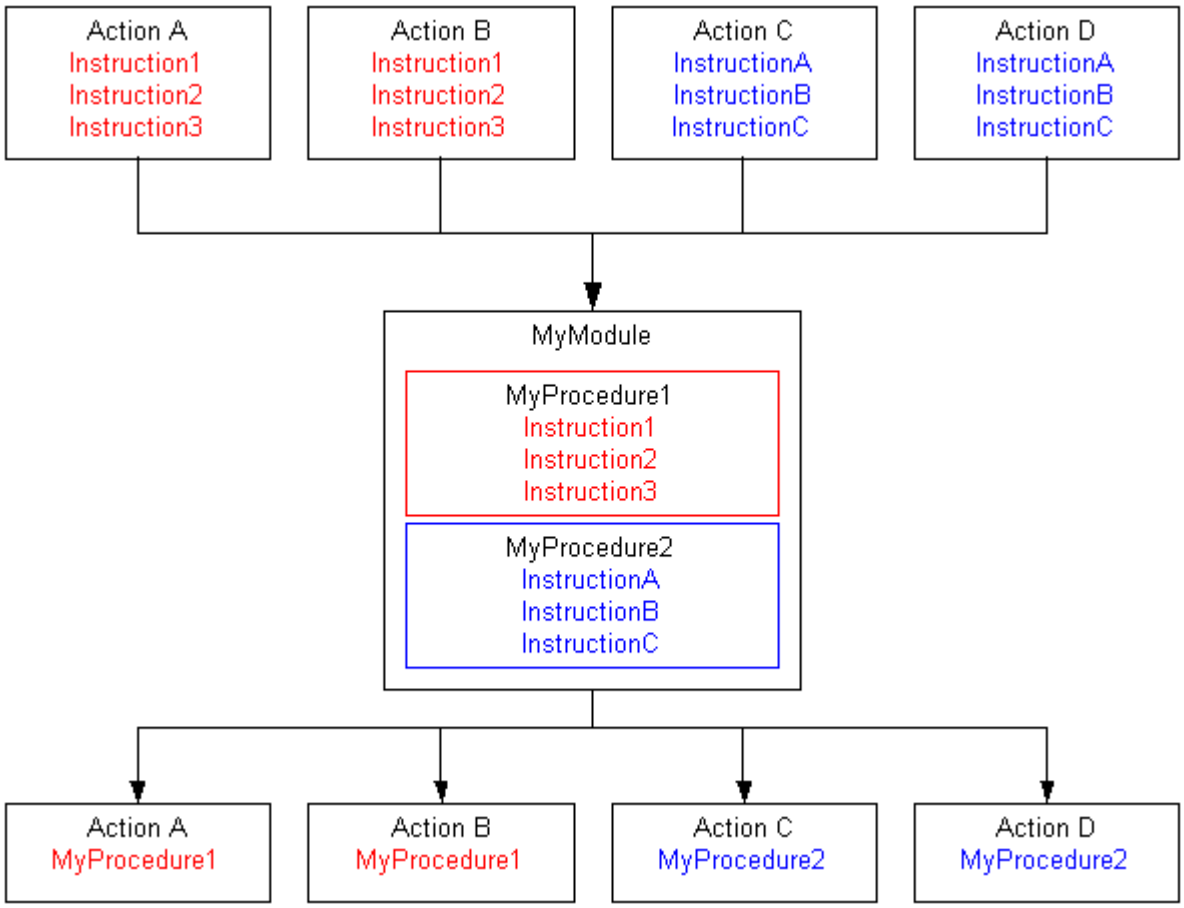
- Procedures: Codes are stored in procedures which are then used at several points in the configuration. Retrieve the code or another procedure in an action by invoking the procedure name. Procedures can be created in WinCC with or without return values. Procedures do not have their own trigger, they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



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.

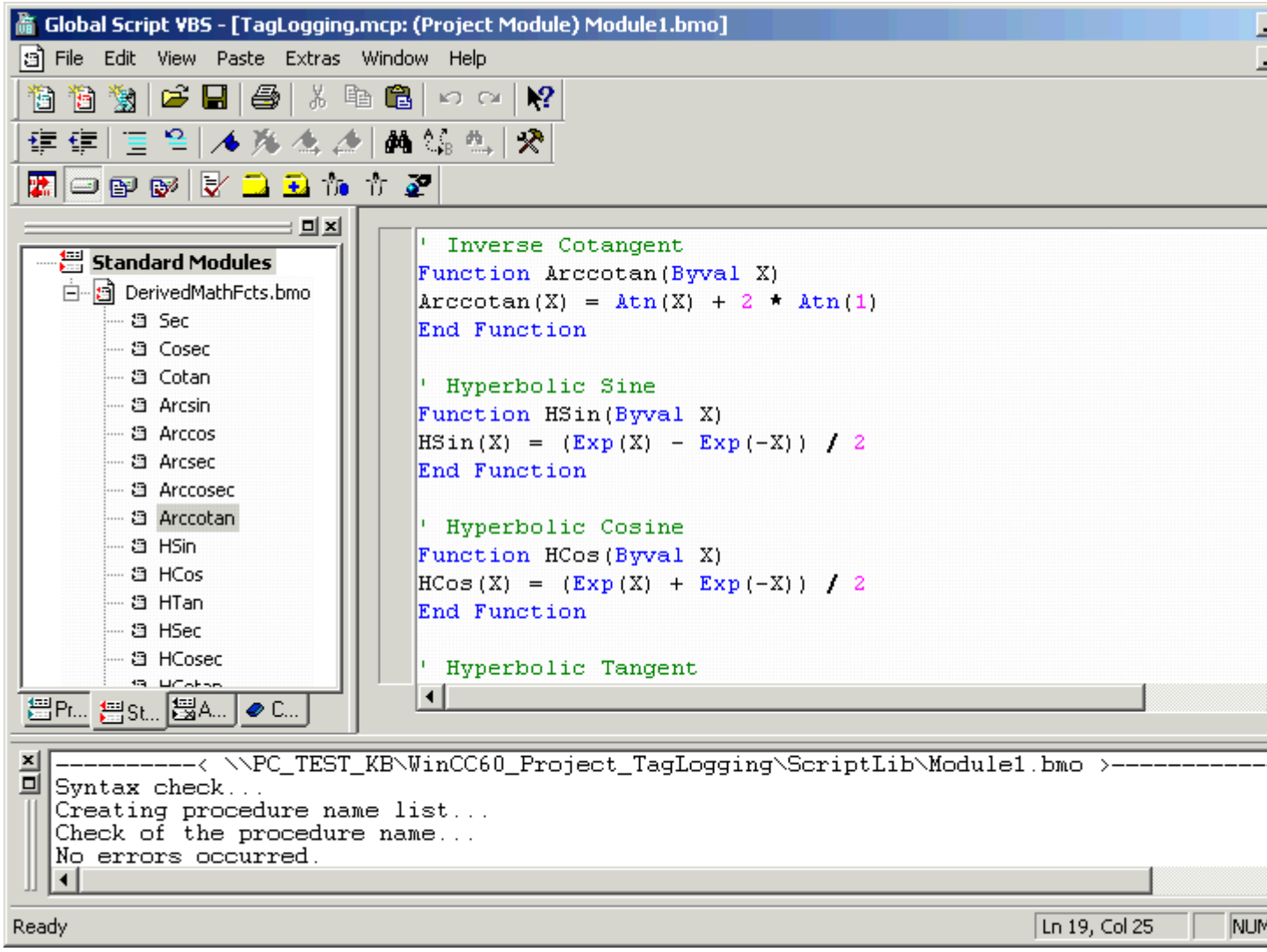


Related procedures should be stored in modules. When a certain procedure is retrieved via an action in Runtime, the module that contains that procedure is loaded. Observe the following when structuring the modules and procedures:

- 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 39)

VBScript Editors (Page 28)

Basic Principles of VBScript (Page 760)

Actions (Page 21)

Using Visual Basic Script in WinCC (Page 14)

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 55)
Basic Principles of VBScript (Page 760)
Modules and Procedures (Page 18)
Using Visual Basic Script in WinCC (Page 14)

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 then 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>\ScriptAct\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 18)
Renaming a Procedure or Module (Page 54)
Saving a Procedure (Page 52)
Protecting a Module with a Password (Page 51)
Adding Module-related Information (Page 49)
Using Standard and Project Procedures (Page 47)
How to Write Procedure Codes (Page 45)
Creating a New Procedure (Page 42)
Creating and Editing Procedures (Page 39)

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 context 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" 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.

Tags and CrossReference

All the pictures which are addressed with the standard formulation

```
' VBS1
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.

Pictures and CrossReference

All the pictures which are addressed with the standard formulation

```
'VBS2
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

VBS Reference (Page 109)

VBScript Editors (Page 28)

Basic Principles of VBScript (Page 760)

Actions (Page 21)

Modules and Procedures (Page 18)

Using Visual Basic Script in WinCC (Page 14)

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. Since event triggered and cyclic/tag triggered actions are separately processed 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 760)

Structure of VBScript Files (Page 97)

Creating and Editing Actions (Page 55)

Creating and Editing Procedures (Page 39)

VBScript Editors (Page 28)

Use of CrossReference (Page 25)

Actions (Page 21)

Modules and Procedures (Page 18)

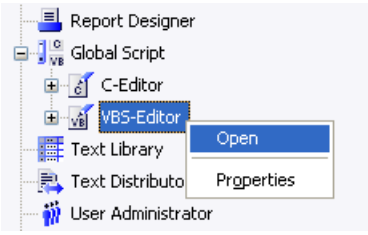
1.8 VBScript Editors

1.8.1 VBScript Editors

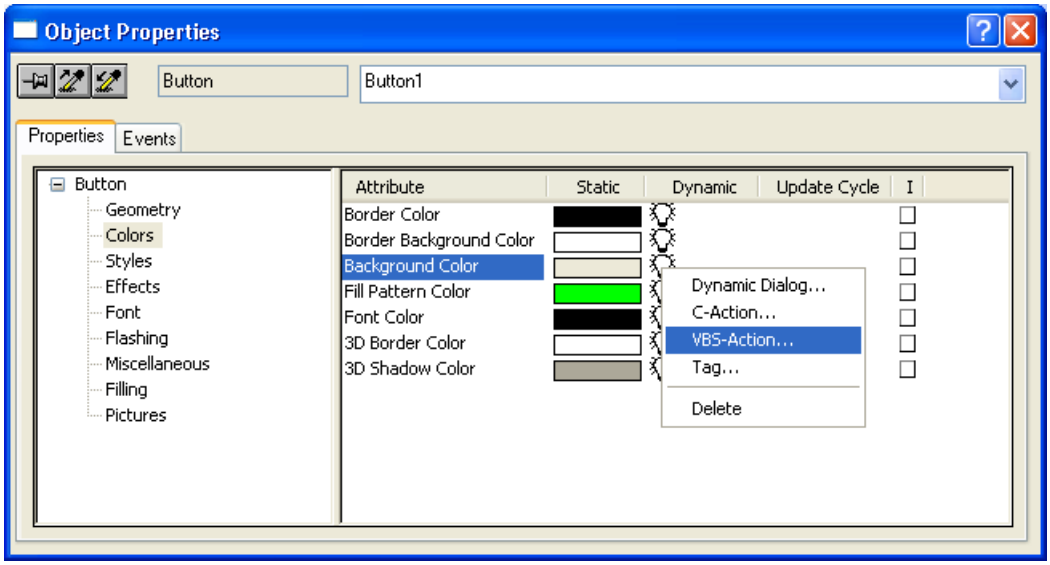
Introduction

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 30)

1.8.2 Global Script Editor

Introduction

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

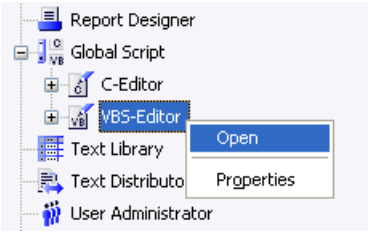
Global Script provides a similar range of functions to that of the C script editor in WinCC.

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 context menu "Open" command in the WinCC Explorer project window.

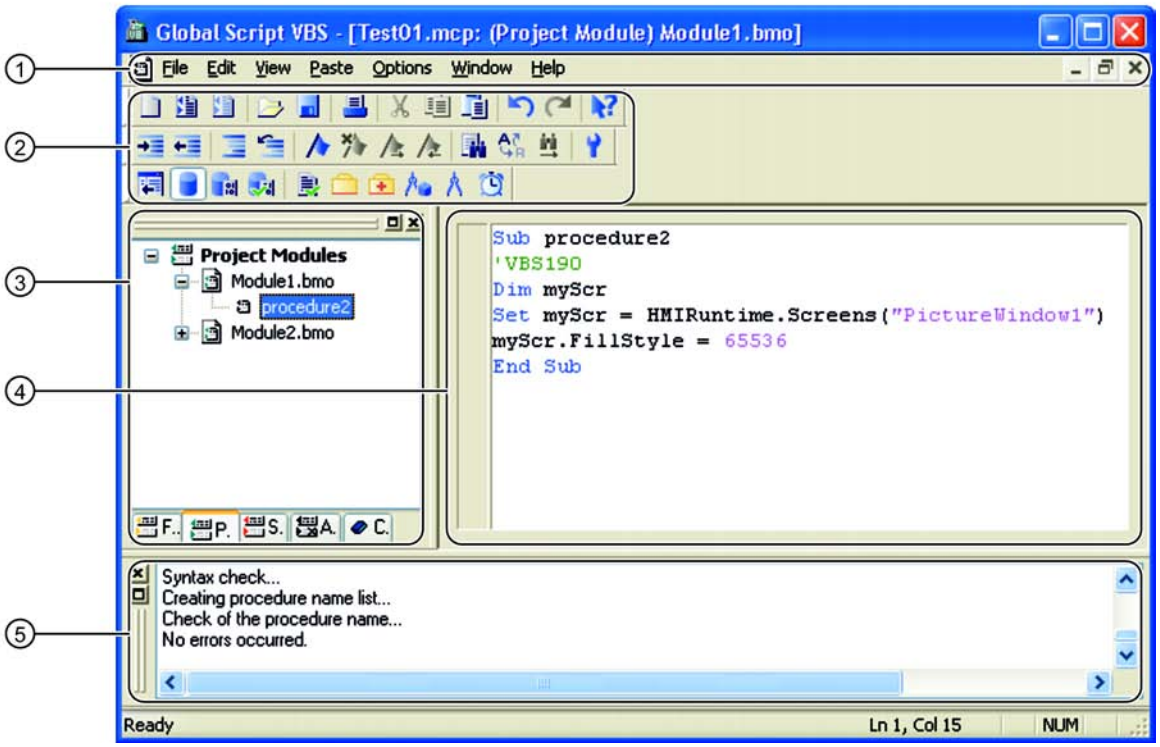


Global Script is also automatically started when a module or action is opened by means of a double click in WinCC Explorer.

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.

See also

Deleting Actions or Procedures (Page 38)

Working with the Toolbars (Page 36)

Working in an Editing Window (Page 32)

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:

Color	Description	Example:
blue	Key words Functions	Sub, End Sub, Next
green	Comments	' is a comment
red	Strings (character strings and digits)	"Object1"
dark blue	Preprocessor statements	--

Color	Description	Example:
bold black	Constants	**vbTrue, vbFalse**
black	Other codes	--

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:

```
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" context menu command in the editing window to display a list of general VBS functions.

Lists of Objects, Properties and Methods

Using the context 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" context menu command to call in a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> but 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 codes by means of 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.

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 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.

Bookmarks

Bookmarks can be set in the code via the icons in the toolbar in order to find certain parts in the codes 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 97)

Global Script Editor (Page 30)

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

Button	Function	Key combination
	Creates a new project module (Global Script only)	<ALT+F1>
	Creates a new standard module (Global Script only)	<ALT+F2>
	Creates a new global action (Global Script only)	<ALT+F3>
	Opens an existing action or existing module (Global Script only)	<CTRL+O>
	Saves the content of the active editing window. This function is only available, when an editing window is open. After saving the display in the navigation window is refreshed. (Global Script only)	<CTRL+S>
	Cuts the selected text and copies it to the clipboard. This function is only available, when text is selected.	<CTRL+X>
	Copies the selected text to the clipboard. This function is only available, when text is selected.	<CTRL+C>
	Pastes the contents of the clipboard at the location of the cursor. This function is only available, when the clipboard is not empty.	<CTRL+V>
	Prints the contents of the active editing window as project documentation. This function is only available, when an editing window is open.	<CTRL+P>

"Editor" Toolbar Content

Button	Function	Key combination
	Indents the line, in which the cursor is located, one position to the right.	--
	Indents the line, in which the cursor is located, one position to the left.	--
	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.	--

Button	Function	Key combination
	Removes the comment marking from the lines selected by the mouse. If no lines have been selected by the mouse, the comment marking in the line in which the cursor is located is removed.	--
	Sets a bookmarks in the current line. Actuating again removed the bookmark from the current line.	<CTRL+F9>
	Removes all bookmarks from the current code in the editing window.	<CTRL+SHIFT+F9>
	Moves the cursor one bookmark further.	<F9>
	Moves the cursor one bookmark back.	<SHIFT+F9>
	Opens the "Find" dialog for text search in the code.	<CTRL+F>
	Opens the "Replace" dialog for search and replace in the code.	<CTRL+H>
	Repeats the search process.	<F3>
	Opens the "Script editor options" dialog.	--
	Undoes the last of a maximum of 30 editor actions. This function is only available, when an editor action has been executed.	<CTRL+Z>
	Redoes the last editor action that was undone. This function is only available, when an editor action has been undone.	<CTRL+Y>

Content of the "Edit" Toolbar

Button	Function	Key combination
Workplace		
	Selects the file in the navigation window to which the current editing window belongs (Global Script only).	--
	Displays all the files in the navigation window (Global Script only).	--
	Only displays the Script files in the navigation window (Global Script only).	--
	Only displays the syntactically correct files in the navigation window (Global Script only).	--
Script		
	Executes a Syntax Check in the code of the current editing window.	<F7>
WinCC Objects		
	Opens the "Info/Trigger" dialog.	<CTRL+T>
	Displays the trigger name.	
	Opens a tag selection dialog returns the selected tag name as the return value.	<CTRL+U>

Button	Function	Key combination
	Opens a tag selection dialog and returns the tag name with an associated reference.	<CTRL+W>
	Opens a picture/object browser in which a picture/ object can be selected whose name is then used for the return value.	<CTRL+Q>
	Opens a picture selection dialog for pictures and returns the picture name, with the server prefix if necessary.	<CTRL+B>

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 30)

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 21)

Modules and Procedures (Page 18)

Global Script Editor (Page 30)

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. The 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 point 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.

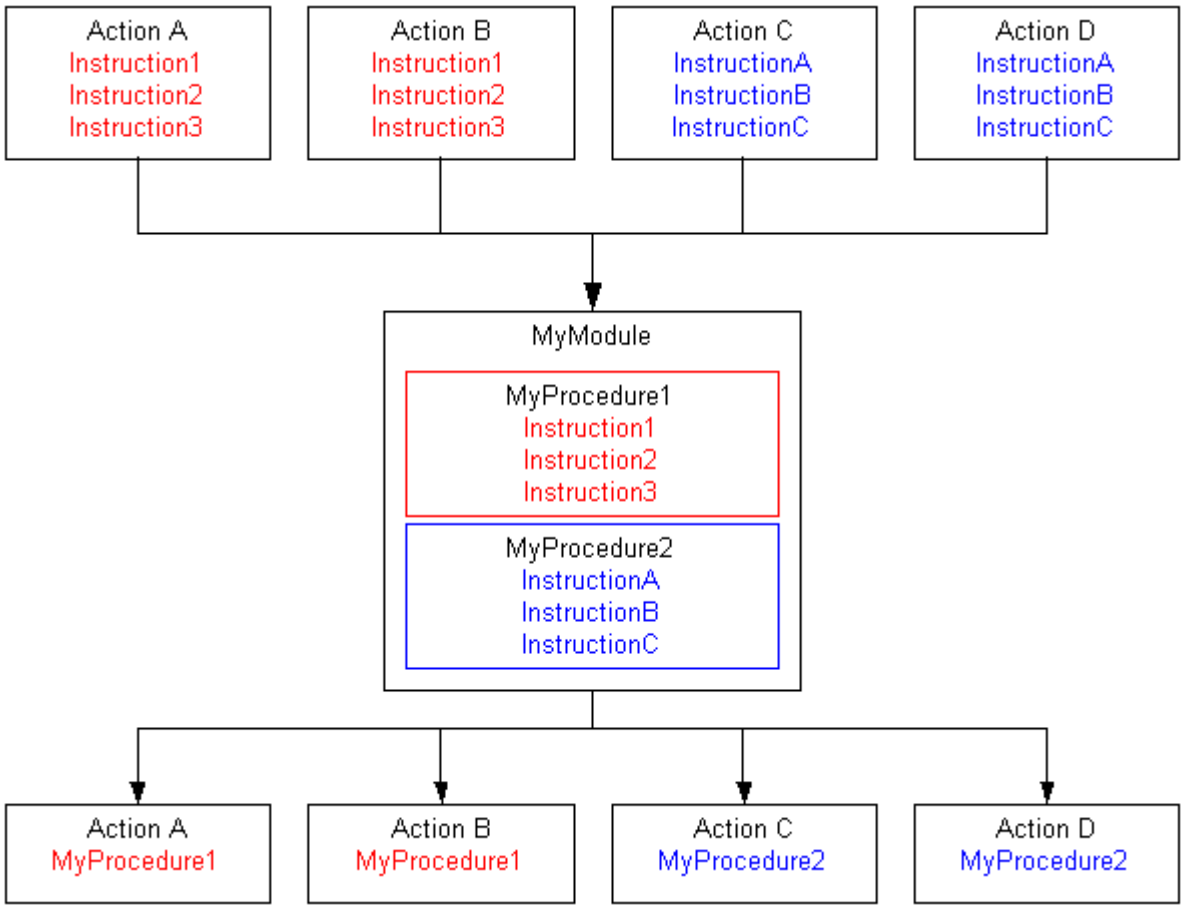
Procedures are created for repeated functionalities, e.g.

- 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.

If you change a standard procedure into a project procedure, and vice versa, you must save the pictures again which use this procedure.

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.

NOTICE
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 o 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 23)

Renaming a Procedure or Module (Page 54)

Saving a Procedure (Page 52)

Protecting a Module with a Password (Page 51)

Adding Module-related Information (Page 49)

Using Standard and Project Procedures (Page 47)

How to Write Procedure Codes (Page 45)

Creating a New Procedure (Page 42)

Examples of VBScript (Page 734)

Modules and Procedures (Page 18)

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 continuos 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.

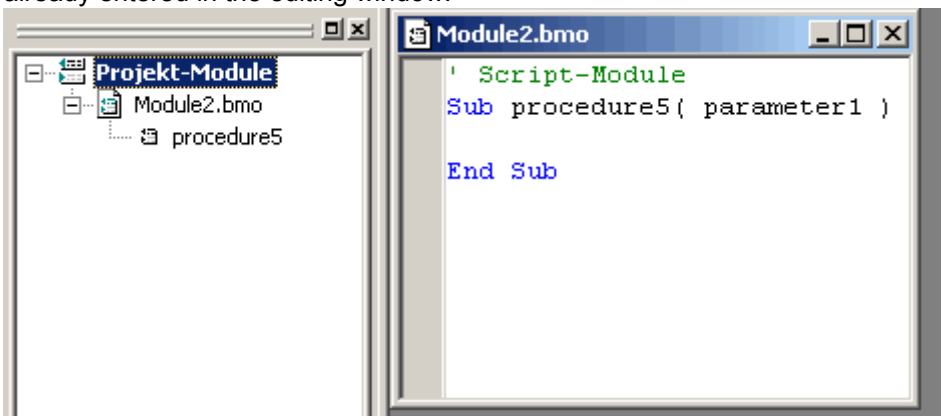
Global procedures (valid for the for entire project) can only be programmed Global Script. Procedures can be called via actions in Graphics Designer and picture-related procedures created in the declaration area of an action. Using a global action in Global Script, it is possible to create procedures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

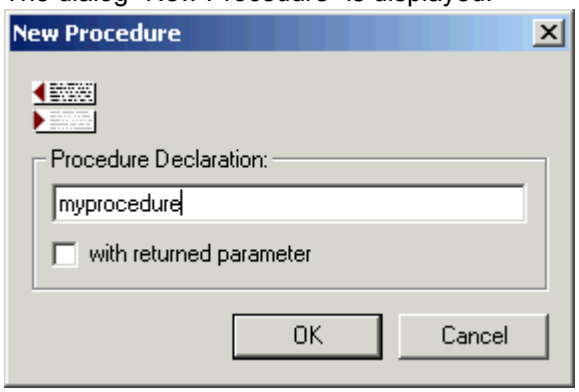
The following procedure describes the creation of a new procedure in Global Script:

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.
   The dialog "New Procedure" is displayed:



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 39)
Multiple Use of Procedures and Actions (Page 23)
Renaming a Procedure or Module (Page 54)
Saving a Procedure (Page 52)
Protecting a Module with a Password (Page 51)
Adding Module-related Information (Page 49)
Using Standard and Project Procedures (Page 47)
How to Write Procedure Codes (Page 45)
Modules and Procedures (Page 18)

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, e.g. to obtain information that the procedure has been executed successfully (e.g. 0 = fault, 1 = successfully 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.

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 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, you are only offered a list of standard properties.

Example of full intellisense:

```
Dim Variable
Set Variable = ScreenItems ("Kreis1")

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" context menu command in the editor window to display a list of general VBS functions.

Lists of Objects, Properties and Methods

Using the context 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" context menu command to call in a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> but 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 codes by means of 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.

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 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. You should therefore also always check your scripts in the Runtime environment.

Changing a Procedure

If a procedure is modified while Runtime is active, the modification becomes effective 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 45)

Multiple Use of Procedures and Actions (Page 23)

Renaming a Procedure or Module (Page 54)

Saving a Procedure (Page 52)

Protecting a Module with a Password (Page 51)

Adding Module-related Information (Page 49)

Creating a New Procedure (Page 42)

Modules and Procedures (Page 18)

Creating and Editing Procedures (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. I 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 teh procedure.
4. Mark the procedure in the navigation window with the mouse.
5. Choose pop-up menu command "Transfer Procedure Retrieval".

See also

Creating and Editing Procedures (Page 39)

Multiple Use of Procedures and Actions (Page 23)

Renaming a Procedure or Module (Page 54)

Saving a Procedure (Page 52)

Protecting a Module with a Password (Page 51)

Adding Module-related Information (Page 49)

How to Write Procedure Codes (Page 45)

Creating a New Procedure (Page 42)

Modules and Procedures (Page 18)

1.9.5 Adding 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:

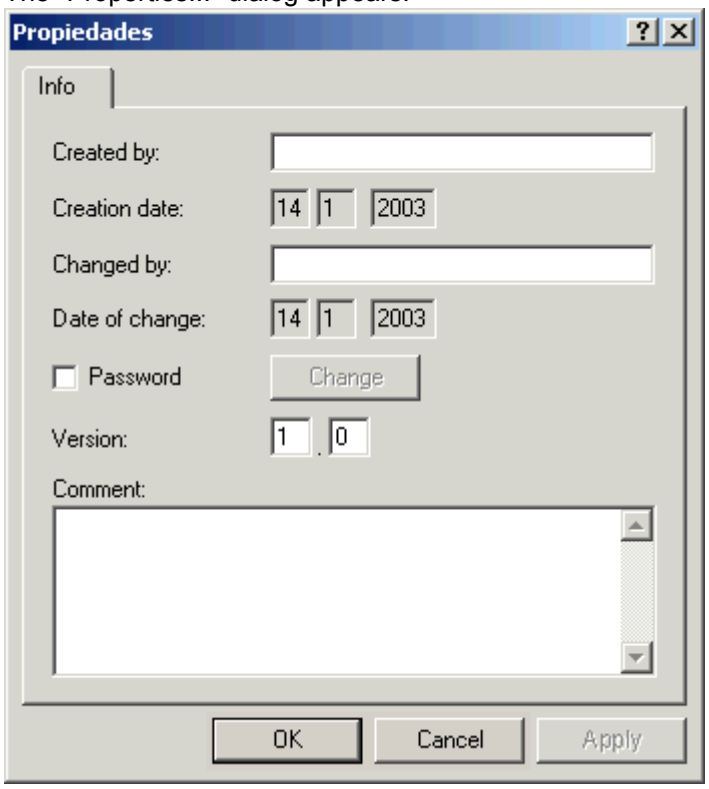
- "Created by"
- "Changed by"
- "Comments" e.g. module functionality/procedures contained

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



4. 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 23)
Renaming a Procedure or Module (Page 54)
Saving a Procedure (Page 52)
Adding Module-related Information (Page 49)
Using Standard and Project Procedures (Page 47)
How to Write Procedure Codes (Page 45)
Creating a New Procedure (Page 42)
Modules and Procedures (Page 18)
Creating and Editing Procedures (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.

NOTICE
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 23)
Renaming a Procedure or Module (Page 54)
Saving a Procedure (Page 52)
Adding Module-related Information (Page 49)
Using Standard and Project Procedures (Page 47)
How to Write Procedure Codes (Page 45)
Creating a New Procedure (Page 42)
Modules and Procedures (Page 18)
Creating and Editing Procedures (Page 39)

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.

NOTICE
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.

Requirements

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

Procedure

1. Click the button "Syntax Check" 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 "Save" in the toolbar.

See also

Diagnostics (Page 83)

Multiple Use of Procedures and Actions (Page 23)

Saving a Procedure (Page 52)

Protecting a Module with a Password (Page 51)

Adding Module-related Information (Page 49)

Using Standard and Project Procedures (Page 47)

How to Write Procedure Codes (Page 45)

Creating a New Procedure (Page 42)

Modules and Procedures (Page 18)

Creating and Editing Procedures (Page 39)

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 23)
Saving a Procedure (Page 52)
Protecting a Module with a Password (Page 51)
Adding Module-related Information (Page 49)
Using Standard and Project Procedures (Page 47)
How to Write Procedure Codes (Page 45)
Creating a New Procedure (Page 42)
Modules and Procedures (Page 18)
Creating and Editing Procedures (Page 39)

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 in 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".

NOTICE

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.

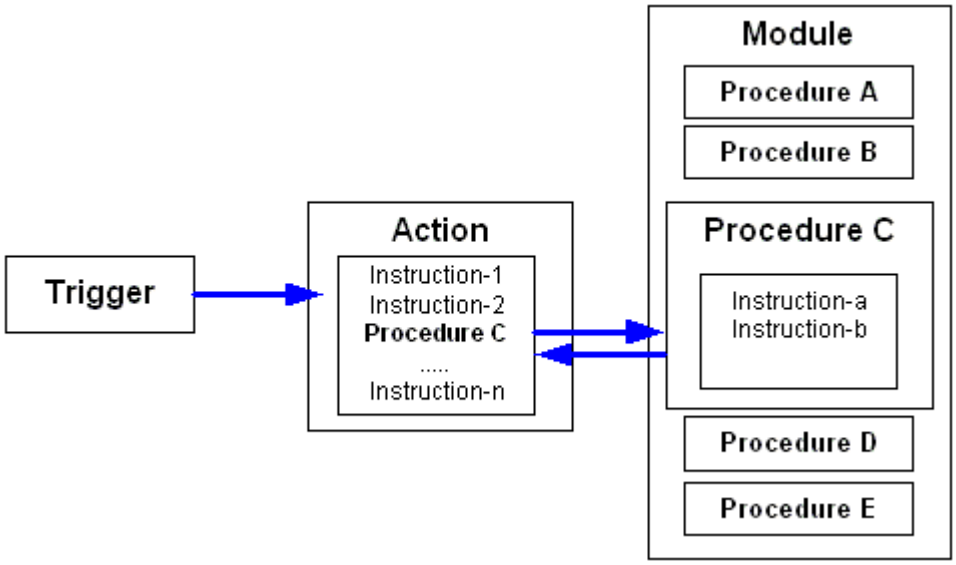
It is also possible to display all the tags and pictures used in actions by means of the WinCC CrossReference. CrossReference can also be used for the to link tag connections of Graphics Designer actions easily.

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 97)

Action and Procedure Names in the Debugger (Page 99)

How to Rename an Action (Page 81)

Saving Actions (Page 67)

Protecting an Action with a Password (Page 66)

How to add information to the action (Page 64)

How to Edit Actions (Page 61)

Creating a New Action (Page 59)

Triggers (Page 69)

Actions (Page 21)

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 ways 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 the adjacent button in the tool bar 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 81)

Saving Actions (Page 67)

Protecting an Action with a Password (Page 66)

How to add information to the action (Page 64)

How to Edit Actions (Page 61)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

Actions (Page 21)

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 while Runtime is active, 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's navigation window and dropping 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 ("Kreis1")

Variable.<Intellisense>
```

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" context menu command in the editor window to display a list of general VBS functions.

Lists of Objects, Properties and Methods

Using the context 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" context menu command to call in a list of possible properties and methods.

The same lists can be called in with the key combination <CTRL + SPACEBAR> but 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 codes by means of 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.

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 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 27)
Testing with the Debugger (Page 90)
Structure of VBScript Files (Page 97)
How to Rename an Action (Page 81)
Saving Actions (Page 67)
Protecting an Action with a Password (Page 66)
How to Edit Actions (Page 61)
Creating a New Action (Page 59)
Triggers (Page 69)
Creating and Editing Actions (Page 55)
Actions (Page 21)

1.10.4 How to add information to the action

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 the 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

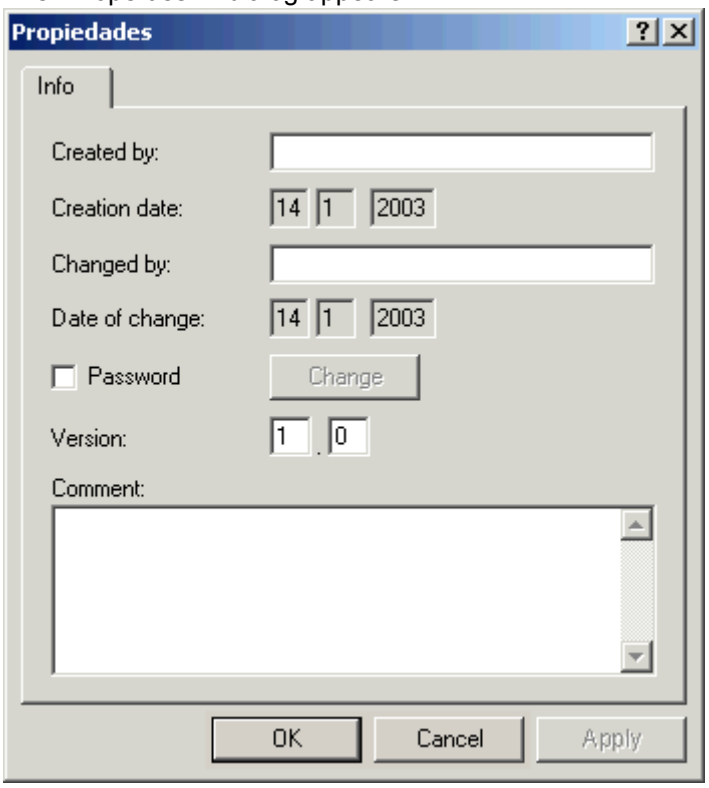
It is also possible to define a password for the action. Further information on assigning passwords is provided in "Protecting Actions with a Password".

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



4. Enter the required information.

See also

How to Rename an Action (Page 81)

Saving Actions (Page 67)

Protecting an Action with a Password (Page 66)

How to Edit Actions (Page 61)

Creating a New Action (Page 59)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

Actions (Page 21)

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.

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

See also

How to Rename an Action (Page 81)
Saving Actions (Page 67)
How to add information to the action (Page 64)
How to Edit Actions (Page 61)
Creating a New Action (Page 59)
Triggers (Page 69)
Creating and Editing Actions (Page 55)
Actions (Page 21)

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 the button "Syntax Check" 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 "Save" in the toolbar.

See also

Actions (Page 21)

How to Rename an Action (Page 81)

Protecting an Action with a Password (Page 66)

How to add information to the action (Page 64)

How to Edit Actions (Page 61)

Creating a New Action (Page 59)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

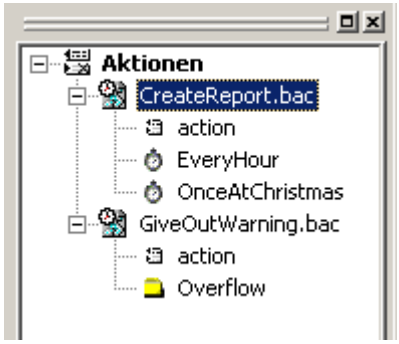
1.10.7 Triggers

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 "" 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.

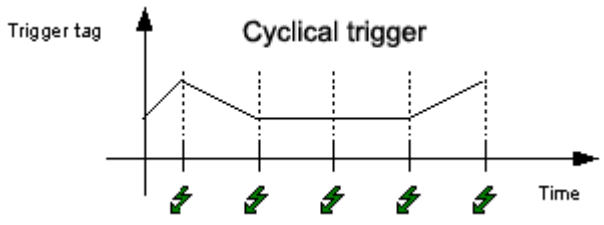
Note

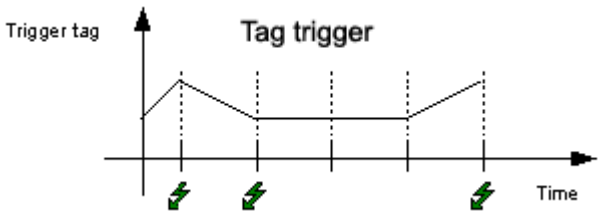
If the action should not be executed at each event, it is possible to define a condition in the action, the result of which controls whether the action is executed or not.

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.

Note

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

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.

Further information on CrossReference is available in the WinCC documentation.

See also

Actions (Page 21)

How to delete a trigger (Page 79)

How to change a trigger (Page 78)

How to add a trigger of the type "Tag" (Page 76)

How to add a trigger of the type "Timer" (Page 74)

Creating and Editing Actions (Page 55)

1.10.7.2 Animation trigger

Animation trigger

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. The following cycles are possible:

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

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.

With a VBS 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 the VBS action in the "Left" property:

```
item.Left = item.Left + 5
```

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

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

```
Dim obj
Set obj = ScreenItems.Item("Rectangle1")
obj.ActivateDynamic "Left", "CycleTime1s"
```

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

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

Note

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

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

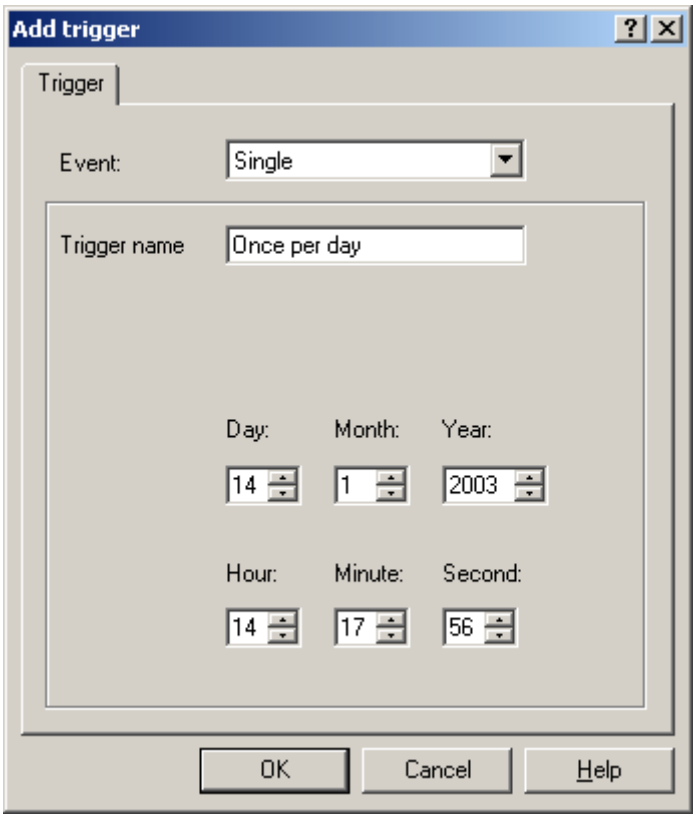
Cyclic triggers guarantee a high updating rate of the system but require high system loads. Choose cyclic triggers only for those actions, where the update is very important. With high system loads, some actions may not be executable.

"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 control.
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 79)

How to add a trigger of the type "Tag" (Page 76)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

Actions (Page 21)

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:

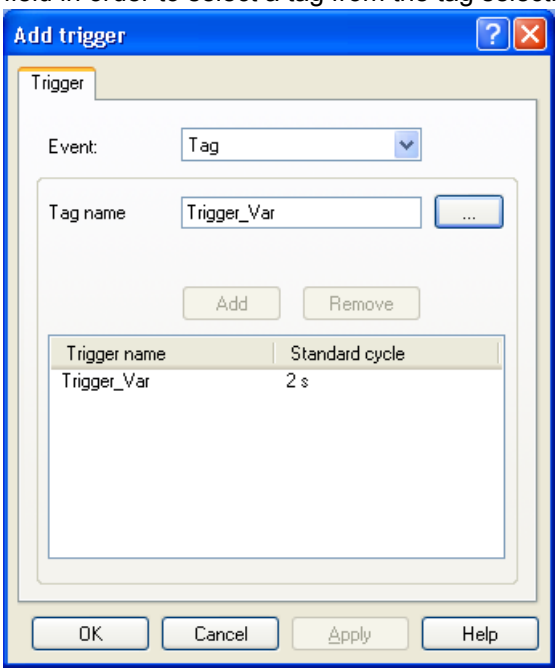
- On change of tag: The action is executed each time the tag value changes. Since this setting causes a very high system utilization, the updating rate should be set as low as possible.
- Query the tag status according to standard cycle (including user cycles): Define a cycle in whose intervals the tag 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.

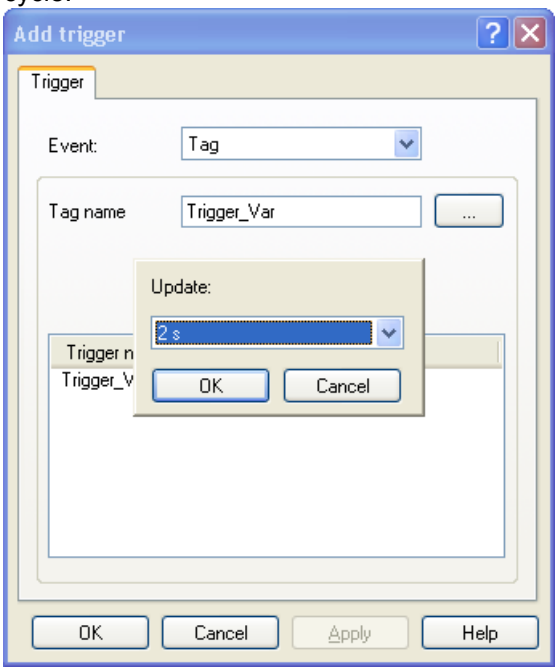
Procedure

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 control.
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.

See also

How to delete a trigger (Page 79)

How to add a trigger of the type "Tag" (Page 76)

How to add a trigger of the type "Timer" (Page 74)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

Actions (Page 21)

1.10.7.5 How to change a trigger

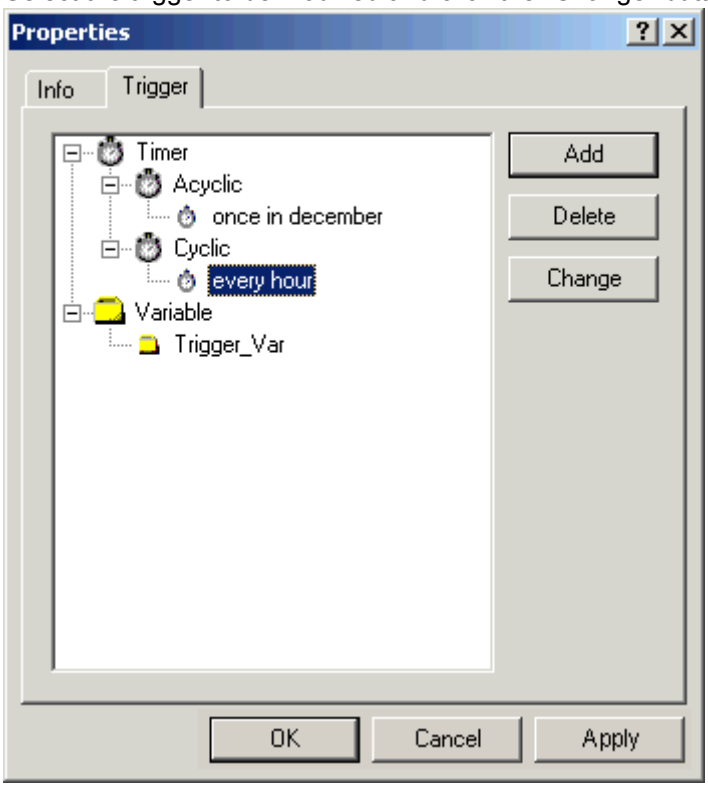
Introduction

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 control.

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 69)

How to delete a trigger (Page 79)

How to add a trigger of the type "Tag" (Page 76)

How to add a trigger of the type "Timer" (Page 74)

Creating and Editing Actions (Page 55)

Actions (Page 21)

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.

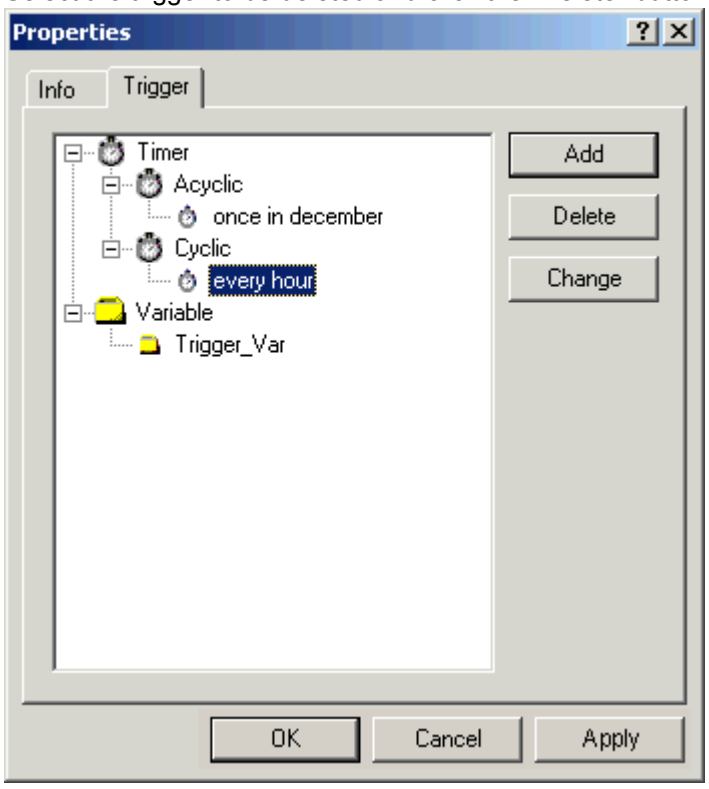
If a trigger is deleted during Runtime, it takes effect after saving the action.

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 button "Info/Trigger" in the toolbar or choose the pop-up menu command "Info". The "Properties..." dialog appears.
4. Select the "Triggers" tab control.
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 21)

How to change a trigger (Page 78)

How to add a trigger of the type "Tag" (Page 76)

How to add a trigger of the type "Timer" (Page 74)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

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 66)

Saving Actions (Page 67)

How to add information to the action (Page 64)

How to Edit Actions (Page 61)

Creating a New Action (Page 59)

Triggers (Page 69)

Creating and Editing Actions (Page 55)

Actions (Page 21)

1.11 Activating Global Actions in Runtime

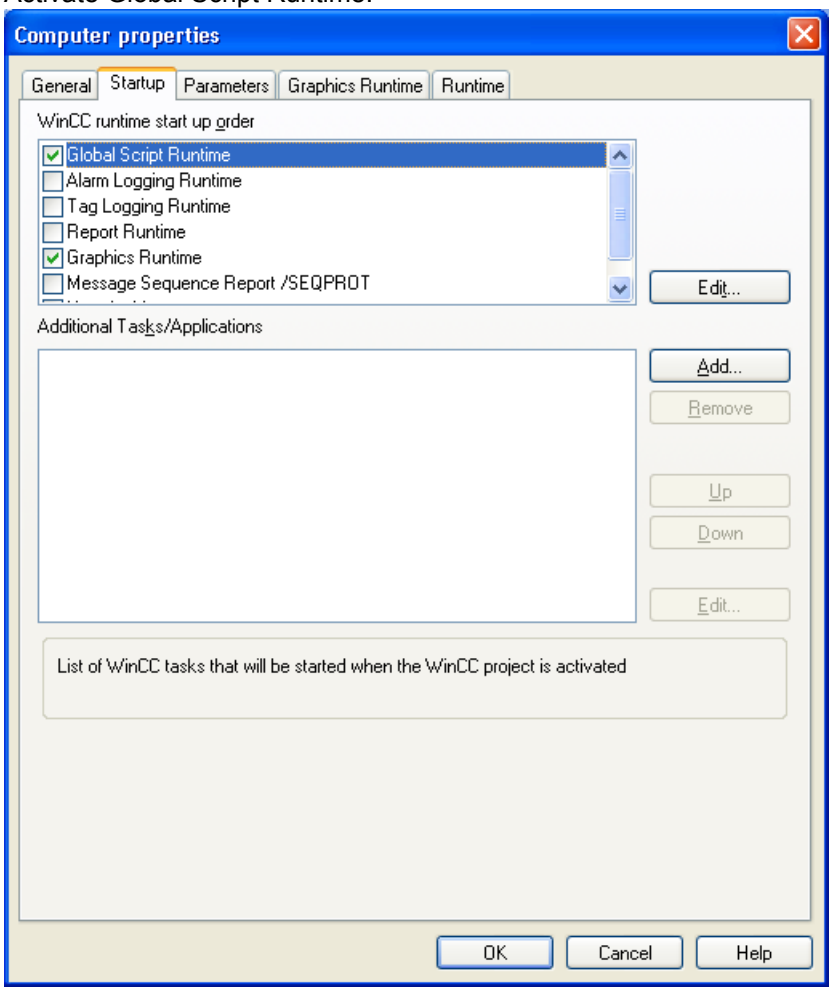
Introduction

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. Activate Global Script Runtime.



4. Click OK to confirm your entries.

See also

Creating and Editing Actions (Page 55)

Creating and Editing Procedures (Page 39)

VBScript Editors (Page 28)

Using Visual Basic Script in WinCC (Page 14)

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 GCS 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 "HMIRuntime.Trace(<Ausgabe>)".

The GSC Diagnostics displays trace output from C and VBS.

Debugger

In order to test the scripts in Runtime, a debugger can be used instead of the Diagnostics window. The utilization of the Microsoft Script Debugger is described in chapter "Testing with the Debugger".

The Microsoft Script Debugger is located in the Microsoft Download-Center under the following URL:

- http://www.microsoft.com/downloads/Search.aspx?displaylang=en (http://www.microsoft.com/downloads/Search.aspx?displaylang=en)

Use the "Search" field to search for "Script Debugger" and select the required download.

See also

Testing with the Debugger (Page 90)

GSC Runtime (Page 87)

GSC Diagnostics (Page 84)

Microsoft Download Center (http://www.microsoft.com/downloads/Search.aspx?displaylang=en)

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 86)

GSC Diagnostics Attributes (Page 86)

Inserting the GSC Diagnostics Window into a Picture (Page 85)

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 86)

GSC Diagnostics Attributes (Page 86)

GSC Diagnostics (Page 84)

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 86)

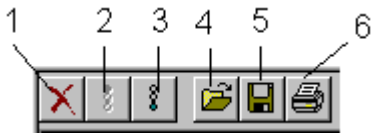
Inserting the GSC Diagnostics Window into a Picture (Page 85)

GSC Diagnostics (Page 84)

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 86)

Inserting the GSC Diagnostics Window into a Picture (Page 85)

GSC Diagnostics (Page 84)

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

Inserting the GSC Runtime Window into a Picture (Page 88)

GSC Runtime Attributes (Page 89)

1.12.3.2 Inserting 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 87)

GSC Runtime Attributes (Page 89)

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 87)

Inserting the GSC Runtime Window into a Picture (Page 88)

1.12.4 Testing with the Debugger

1.12.4.1 Testing with the Debugger

Overview

A debugger can be used to test the VBScripts in Runtime, e.g.:

- Microsoft Script Debugger (runs with Windows 2000 and Windows XP)
- Debugger "InterDev" (contained in scope of installation material supplied with Developer Studio)
- Microsoft Script Editor (MSE) Debugger (contained in material supplied with Microsoft Office)

The following description relates exclusively to handling the Microsoft Script Debugger.

Download the Microsoft Script Debugger

The Microsoft Script Debugger is located in the Microsoft Download-Center under the following URL:

- http://www.microsoft.com/downloads/Search.aspx?displaylang=en833a6a92-961e-4ce1-9069-528d22605127

Use the "Search" field to search for "Script Debugger" and select the required download.

Notes on the MSE Debugger

The following settings must be changed when using the MSE Debugger so that the running processes will be displayed:

1. Select the "Properties" button in the "Processes" window.
2. Activate the option "Just-In-Time-Debugging" in the "Debuffer properties" dialog.
3. Restart the computer.
4. Deactivate the "Disable script debugging" option in the MS Internet Explorer so that the Internet Explorer cannot prevent the WinCC debugging procedure.

See also

Principles of Debugging (Page 93)

Executing Script Commands (Page 107)

How to Determine and Modify Tag and Property Values (Page 106)

How to Set Bookmarks in Scripts (Page 105)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Processing Scripts Step-by-Step (Page 102)

Selecting a Script for Editing (Page 100)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Components of the Microsoft Script Debuggers (Page 95)

How to Activate the Debugger (Page 91)

Diagnostics (Page 83)

Microsoft Download Center (http://www.microsoft.com/downloads/Search.aspx?displaylang=en)

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.

Requirements

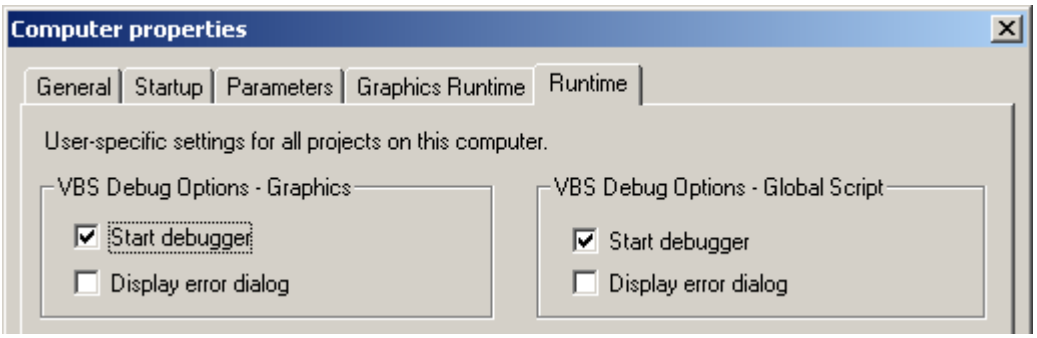
The Microsoft Script Debugger must be installed on the configuration computer.

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 105)
Executing Script Commands (Page 107)
How to Determine and Modify Tag and Property Values (Page 106)
Deleting Breakpoints (Page 104)
Setting Breakpoints (Page 103)
Processing Scripts Step-by-Step (Page 102)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.3 Principles of Debugging

Introduction

The Microsoft Script Debugger can be used to debug the VBScripts. The Microsoft Script 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 90)

Executing Script Commands (Page 107)

How to Determine and Modify Tag and Property Values (Page 106)

How to Set Bookmarks in Scripts (Page 105)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Processing Scripts Step-by-Step (Page 102)

Selecting a Script for Editing (Page 100)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Components of the Microsoft Script Debuggers (Page 95)

How to Activate the Debugger (Page 91)

Diagnostics (Page 83)

1.12.4.4 Components of the Microsoft Script Debuggers

Introduction

The Microsoft Script Debugger offers several components which assist in debugging:

"Command Window"

The "Command Window" is called in using the "View" > "Command Window" menu commands.

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 and properties in the script currently running. Changes made in the "Command Window" are effected directly in the running script so that planned changes can be tested immediately.

The following actions can be executed in the "Command Window":

- Enter commands: Commands can be entered and executed directly in VBScript.
- Change tag values: Tag values can be compiled and modified directly in the "Command Window". This relates to both tags in the current script as well as global tags.
- Modify properties: It is possible to read and write the properties of all objects in the current script context.

The "Command Window" can always be used when a script has reached a breakpoint or a skip has been made from a breakpoint to other commands.

Note

Please note that the changes executed in the "Command Window" have no effect on the source code of the script but only serve for test purposes in the debugger.

"Running Documents" Window

The "Running Documents" window is called via the "View" > "Running Documents" menu command.

This window displays all the scripts currently running in WinCC Runtime, separated according to scripts, from Global Script ("Global Script Runtime") and scripts from the graphical Runtime system ("PDLRT"). All the running Global Script Runtime actions and modules are displayed. In the graphical Runtime system, the scripts are separated according to trigger-controlled actions (picturename_trigger) and event-controlled actions (picturename_events).

"Call Stack" Window

The "Call Stack" window is called via the "View" > "Call Stack" menu command.

This window displays a list of all running actions and called procedures. When a procedure is called, for example, the name is added to the "Call Stack" list. When the procedure has finished, the name is removed from the list. A procedure can be selected from the list in order to skip to the corresponding position in the script document at which the procedure was called.

See also

Executing Script Commands (Page 107)

How to Determine and Modify Tag and Property Values (Page 106)

How to Set Bookmarks in Scripts (Page 105)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Processing Scripts Step-by-Step (Page 102)

Selecting a Script for Editing (Page 100)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Principles of Debugging (Page 93)

How to Activate the Debugger (Page 91)

Testing with the Debugger (Page 90)

Diagnostics (Page 83)

1.12.4.5 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

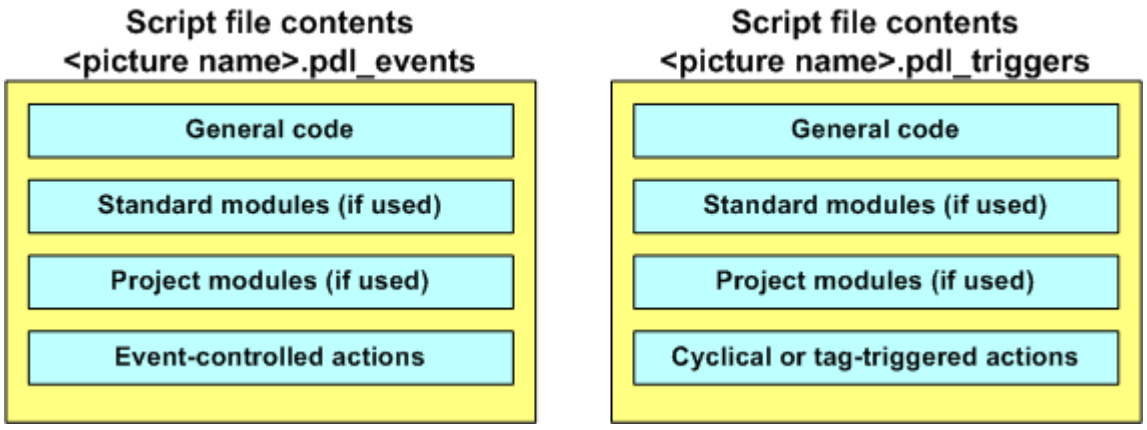
When scripts are debugged with a debugger, the script files always open the different Runtime systems.

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

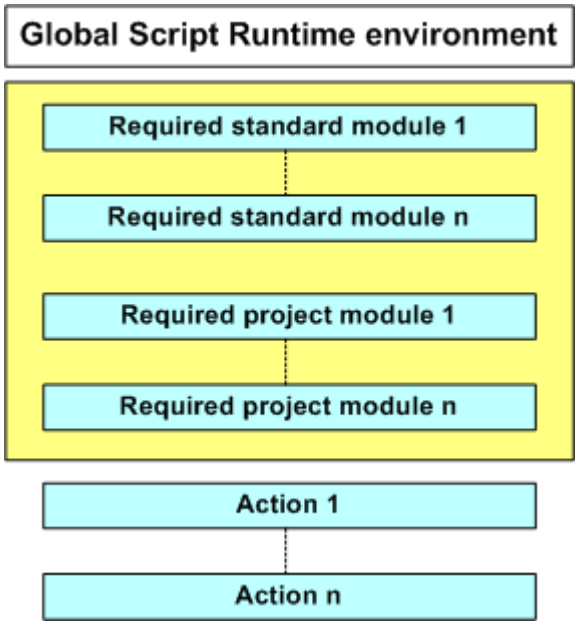
- <Bildname>.pdl_events: Contains the event-driven actions.
- <Bildname>.pdl_triggers: Contains the cyclic and tag-controlled actions.

The following section describes how the script files are structured:

Graphical Runtime system



Global Script Runtime



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 sate described in "Action and Procedure Names in the Debugger".

See also

How to Activate the Debugger (Page 91)
Executing Script Commands (Page 107)
How to Determine and Modify Tag and Property Values (Page 106)
How to Set Bookmarks in Scripts (Page 105)
Deleting Breakpoints (Page 104)
Setting Breakpoints (Page 103)
Processing Scripts Step-by-Step (Page 102)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.6 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:

Action type	Name of the script file
Cyclic or tag-driven actions on a property	ObjectName_PropertyName_Trigger
Mouse events	ObjektName_OnClick ObjektName_OnLButtonDown ObjektName_OnLButtonUp ObjektName_OnRButtonDown ObjektName_OnRButtonUp
Keyboard events	ObjektName_OnKeyDown ObjektName_OnKeyUp
Object events	ObjektName_OnObjectChanged ObjektName_OnSetFocus
Events on properties	ObjektName_PropertyName_OnPropertyChanged ObjektName_PropertyName_OnPropertyStateChanged
Picture events	Document_OnOpen Document_OnClosed

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 107)
How to Determine and Modify Tag and Property Values (Page 106)
How to Set Bookmarks in Scripts (Page 105)
Deleting Breakpoints (Page 104)
Setting Breakpoints (Page 103)
Processing Scripts Step-by-Step (Page 102)
Selecting a Script for Editing (Page 100)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
How to Activate the Debugger (Page 91)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.7 Selecting a Script for Editing

Introduction

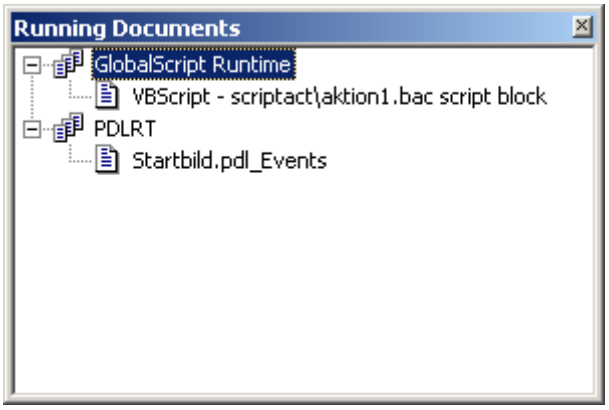
If the Microsoft Script Debugger is called from the Windows Start menu instead of automatic activation using WinCC, scripts which are currently running can be called in for editing in Runtime.

Prerequisite

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

What to do

1. Start the debugger from the Windows Start menu ("Start" > "Programs" > "Options" > "Microsoft Script Debugger").
2. Activate the "View" > "Running Documents" commands from the menu bar.
   The "Running Documents" window is opened. This window displays all the scripts currently running in WinCC Runtime, separated according to scripts, from Global Script ("Global Script Runtime") and scripts from the graphical Runtime system ("PDLRT").



   The example above, "Startbild.pdl" is active and only event-driven scripts are available in the start picture.
3. Double click on the script document in the "Running Documents" window that is to be debugged.
   The script document is opened "read-only" in the Debugger window.

See also

Executing Script Commands (Page 107)

How to Determine and Modify Tag and Property Values (Page 106)

How to Set Bookmarks in Scripts (Page 105)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Processing Scripts Step-by-Step (Page 102)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Components of the Microsoft Script Debuggers (Page 95)

Principles of Debugging (Page 93)

How to Activate the Debugger (Page 91)

Testing with the Debugger (Page 90)

Diagnostics (Page 83)

1.12.4.8 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 93)
Executing Script Commands (Page 107)
How to Determine and Modify Tag and Property Values (Page 106)
How to Set Bookmarks in Scripts (Page 105)
Deleting Breakpoints (Page 104)
Setting Breakpoints (Page 103)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
How to Activate the Debugger (Page 91)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.9 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 104)
Executing Script Commands (Page 107)
How to Determine and Modify Tag and Property Values (Page 106)
How to Set Bookmarks in Scripts (Page 105)
Setting Breakpoints (Page 103)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
How to Activate the Debugger (Page 91)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.10 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 107)
How to Determine and Modify Tag and Property Values (Page 106)
How to Set Bookmarks in Scripts (Page 105)
Setting Breakpoints (Page 103)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
How to Activate the Debugger (Page 91)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

1.12.4.11 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 into the cell where you wish to set a bookmark, and click <CTRL +F2> to set or delete a bookmark.

Skipping to the next bookmark

Press <F2> to skip to the next bookmark in the script.

Skipping to the previous bookmark

Press <SHIFT+F2> to skip to the previous bookmark in the script.

See also

Executing Script Commands (Page 107)

How to Determine and Modify Tag and Property Values (Page 106)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Selecting a Script for Editing (Page 100)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Components of the Microsoft Script Debuggers (Page 95)

Principles of Debugging (Page 93)

How to Activate the Debugger (Page 91)

Testing with the Debugger (Page 90)

Diagnostics (Page 83)

1.12.4.12 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 93)

Executing Script Commands (Page 107)

How to Set Bookmarks in Scripts (Page 105)

Deleting Breakpoints (Page 104)

Setting Breakpoints (Page 103)

Selecting a Script for Editing (Page 100)

Action and Procedure Names in the Debugger (Page 99)

Structure of VBScript Files (Page 97)

Components of the Microsoft Script Debuggers (Page 95)

How to Activate the Debugger (Page 91)

Testing with the Debugger (Page 90)

Diagnostics (Page 83)

1.12.4.13 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 106)
How to Set Bookmarks in Scripts (Page 105)
Deleting Breakpoints (Page 104)
Setting Breakpoints (Page 103)
Selecting a Script for Editing (Page 100)
Action and Procedure Names in the Debugger (Page 99)
Structure of VBScript Files (Page 97)
Components of the Microsoft Script Debuggers (Page 95)
Principles of Debugging (Page 93)
How to Activate the Debugger (Page 91)
Testing with the Debugger (Page 90)
Diagnostics (Page 83)

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 55)

Creating and Editing Procedures (Page 39)

VBScript Editors (Page 28)

Using Visual Basic Script in WinCC (Page 14)

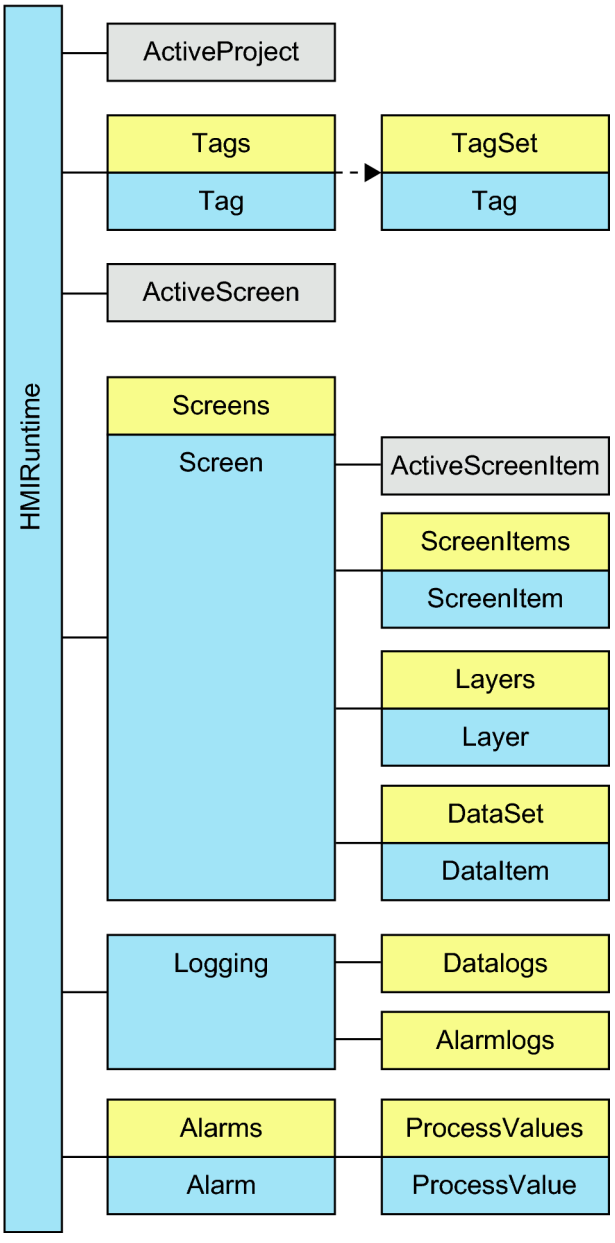
1.14 VBS Reference

1.14.1 VBS Reference

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.



Object and List

Object

Property provided by an object

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

ActiveScreen Property (Page 278)

Object Types of the ScreenItem Object (Page 152)

Methods (Page 638)

Properties (Page 276)

Objects and Lists (Page 113)

AlarmLogs Object (Page 118)

DataItem Object (Page 119)

DataLogs Object (Page 121)

DataSet-Objekt (List) (Page 122)

HMIRuntime Object (Page 125)

Layer Object (Page 127)

Layers Object (Listing) (Page 129)

ScreenItem Object (Page 134)

ScreenItems Object (List) (Page 138)

Screen Object (Page 140)

Screens Object (List) (Page 143)

Tag Object (Page 146)

Tags Object (List) (Page 149)

TagSet Object (List) (Page 151)

ActiveProject Property (Page 278)

ActiveScreenItem Property (Page 279)

Logging Object (Page 130)

Alarm object (Page 115)

Alarms object (list) (Page 116)

ProcessValue Object (Page 131)

ProcessValues Object (List) (Page 132)

1.14.2 Objects and Lists

1.14.2.1 Objects and Lists

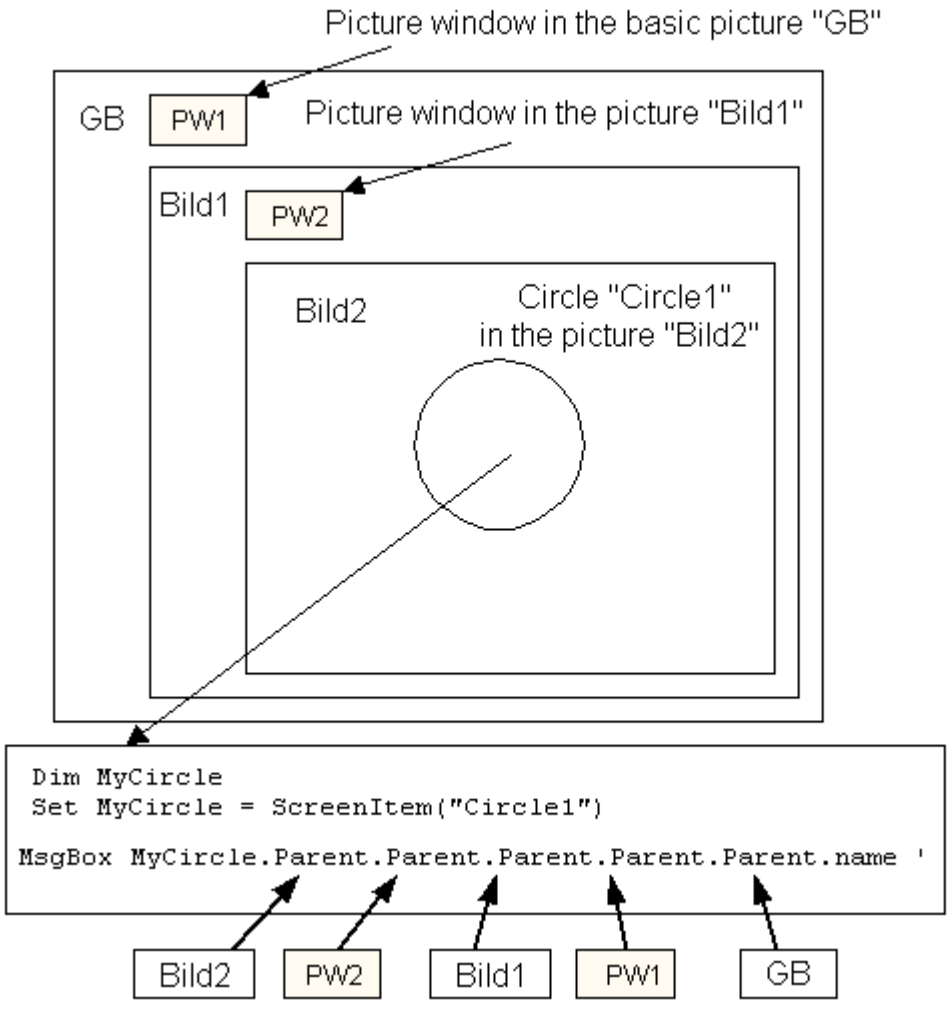
Overview

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:



Only the basic picture name is issued in this 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

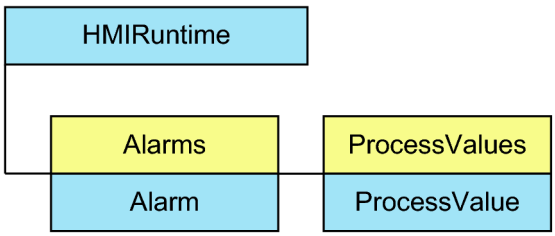
- Tags
- TagSet

See also

ScreenItems Object (List) (Page 138)
TagSet Object (List) (Page 151)
Tags Object (List) (Page 149)
Tag Object (Page 146)
Screens Object (List) (Page 143)
Screen Object (Page 140)
ScreenItem Object (Page 134)
Layers Object (Listing) (Page 129)
Layer Object (Page 127)
Item Object (Page 126)
HMIRuntime Object (Page 125)

1.14.2.2 Alarm object

Description



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

Note

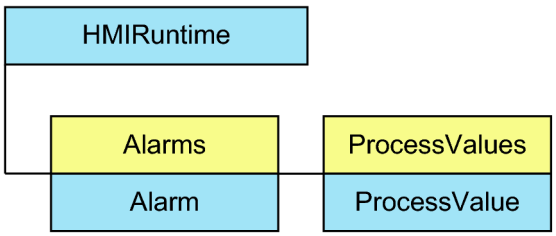
The properties of the alarm object are not automatically updated when the values of the properties change.

See also

Alarms object (list) (Page 116)

1.14.2.3 Alarms object (list)

Description



With the alarm object you can configure 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)
- Read the status of a message (State property)
- Read 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 or set the name of the user who triggered the message (UserName property)
- Read or set the name of the process value blocks (ProcessValues property)
- Read or set the message commentary (Comment property)
- Read or set the message server prefix (Context property)

Example

In the following example, a message is created and configured:

```
'VBS200
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 556)

ComputerName property (Page 345)

Context property (Page 345)

State property (Page 524)

AlarmID property (Page 282)

Instance property (Page 402)

Comment property (Page 344)

UserName property (Page 606)

ProcessValue property (Page 485)

Alarm object (Page 115)

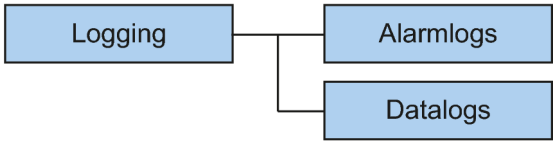
ProcessValues Object (List) (Page 132)

Create method (Page 644)

Item Method (Page 687)

1.14.2.4 AlarmLogs Object

Description



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.

```
'VBS187
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Restore("D:
\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

See also

Error Messages from Database Area (Page 732)

Restore Method (Page 708)

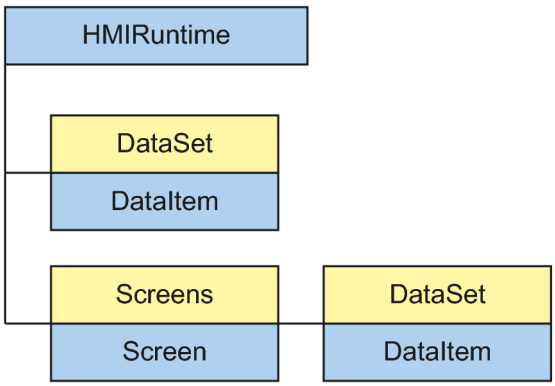
Remove Method (Page 703)

DataLogs Object (Page 121)

Logging Object (Page 130)

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
HMIRuntime.Trace "motor1: " & HMIRuntime.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 & vbNewLine
Next
```

Note

For objects, value may possibly not be output directly

See also

Screen Object (Page 140)

HMIRuntime Object (Page 125)

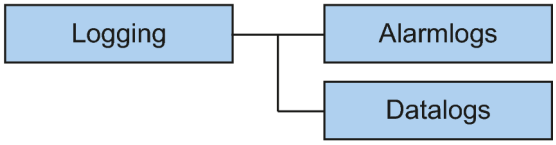
DataSet-Objekt (List) (Page 122)

Value Property (Page 610)

Name Property (Page 452)

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.

```
'VBS188
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:
\Folder","2004-09-14","2004-09-20",-1,1) & vbNewLine
```

See also

Error Messages from Database Area (Page 732)

Restore Method (Page 708)

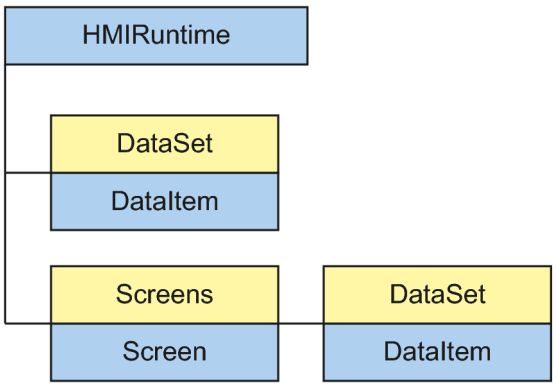
Remove Method (Page 703)

AlarmLogs Object (Page 118)

Logging Object (Page 130)

1.14.2.7 DataSet-Objekt (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 119)

RemoveAll Method (Page 707)

Remove Method (Page 703)

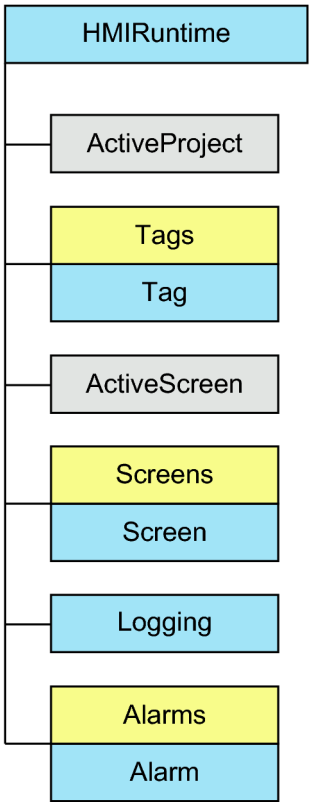
Item Method (Page 687)

Count Property (Page 346)

Add Method (Page 641)

1.14.2.8 HMIRuntime Object

Description



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 143)
TagSet Object (List) (Page 151)
Tags Object (List) (Page 149)
Logging Object (Page 130)
DataSet-Objekt (List) (Page 122)
Visible Property (Page 624)
Trace Method (Page 724)
Tags Property (Page 531)
Stop Method (Page 723)
AlignmentLeft Property (Page 283)
Logging Property (Page 431)
Language Property (Page 406)
DataSet Property (Page 351)
CurrentContext Property (Page 347)
BaseScreenName Property (Page 302)
ActiveProject Property (Page 278)
ActiveScreen Property (Page 278)
MenuToolBarConfig Property (Page 442)
Alarms object (list) (Page 116)

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:

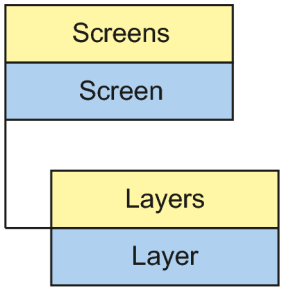
```
'VBS195
Item.BackColor = RGB(255,0,0)
```

See also

Objects and Lists (Page 113)

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:

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

See also

Layer Object (Page 127)

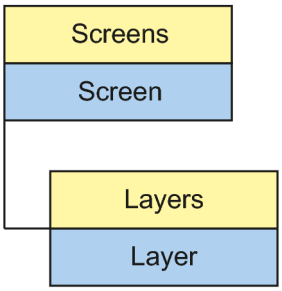
Visible Property (Page 624)

Parent Property (Page 471)

Name Property (Page 452)

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 471)

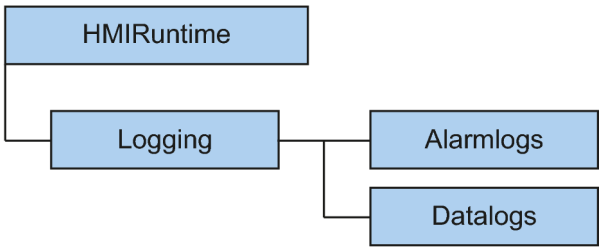
Item Method (Page 687)

Count Property (Page 346)

Layer Object (Page 127)

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) & vbNewLine
```

See also

Error Messages from Database Area (Page 732)

DataLogs Object (Page 121)

AlarmLogs Object (Page 118)

Restore Method (Page 708)

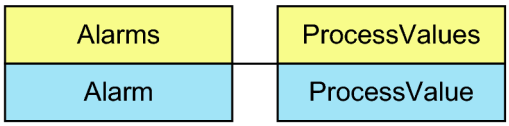
Remove Method (Page 703)

DataLogs Property (Page 351)

AlarmLogs Property (Page 283)

1.14.2.13 ProcessValue Object

Description



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

Note

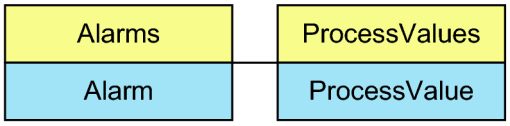
Only the 10 predefined ProcessValues are supported.

See also

ProcessValues Object (List) (Page 132)

1.14.2.14 ProcessValues Object (List)

Description



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 116)

ProcessValue Object (Page 131)

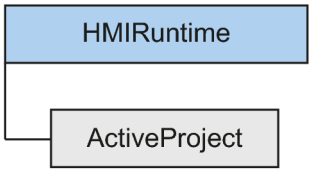
Count Property (Page 346)

Value Property (Page 610)

Item Method (Page 687)

1.14.2.15 Project Object

Description



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

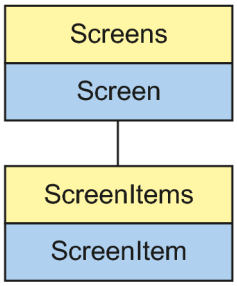
ActiveProject Property (Page 278)

Name Property (Page 452)

Path Property (Page 473)

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:

Standard objects	Smart objects	Windows objects	Tube objects	Controls	Others
Ellipse	3D bar	Button	Double T-piece	Siemens HMI Symbol Library	Customized object
Ellipse arc	Application window	Check box	Polygon tube	WinCC AlarmControl	Group
Ellipse segment	Bar	Radio box	Tube bend	WinCC digital/ analog clock control	
Circle	Picture window	Round button	T-piece	WinCC FunctionTrendControl	
Circular arc	Control	Slider		WinCC gauge control	
Pie segment	I/O field			WinCC OnlineTrendControl	
Line	Faceplate Instance			WinCC OnlineTableControl	
Polygon	Graphic object			WinCC push button control	
Polyline	Combo box			WinCC RulerControl	
Rectangle	List box			WinCC slider control	

Standard objects	Smart objects	Windows objects	Tube objects	Controls	Others
Rounded rectangle	Multiple row text			WinCC UserArchiveControl	
Connector	OLE object				
	Group display				
	Text list				
	Status display				

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.

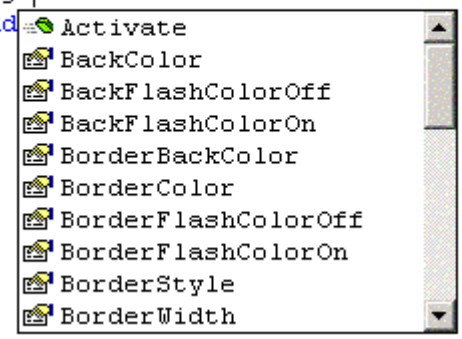
Object properties

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

```
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:

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



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:

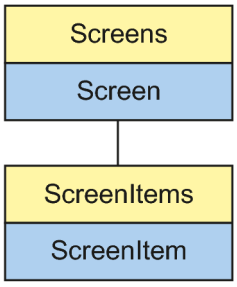
```
Sub OnClick(ByVal Item)
'VBS5
Dim objCircle
Set objCircle= ScreenItems("Circle1")
objCircle.Radius = 2
End Sub
```

See also

Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
Parent Property (Page 471)
Left Property (Page 423)
Layer Property (Page 408)
Height Property (Page 391)
Enabled Property (Page 360)
Activate Method (Page 639)
Example: How to Read Tag Values (Page 741)
Example: Writing tag values (Page 739)
Properties (Page 276)
Objects and Lists (Page 113)
Object Types of the ScreenItem Object (Page 152)

1.14.2.17 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 346)

Example: How to Read Tag Values (Page 741)

Example: Writing tag values (Page 739)

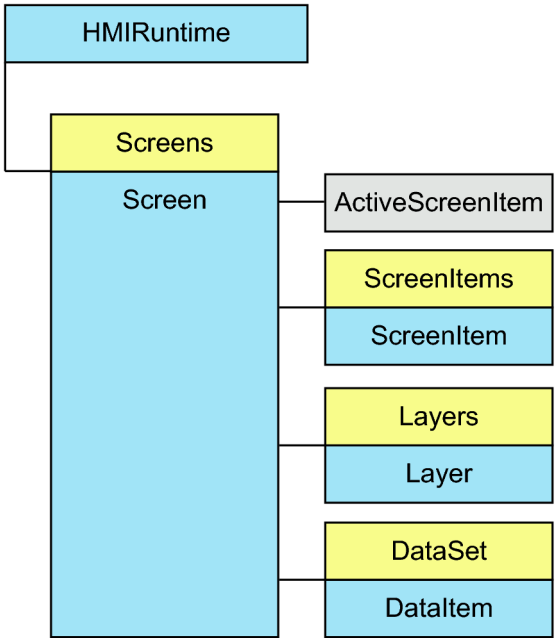
ScreenItem Object (Page 134)

Parent Property (Page 471)

Item Method (Page 687)

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

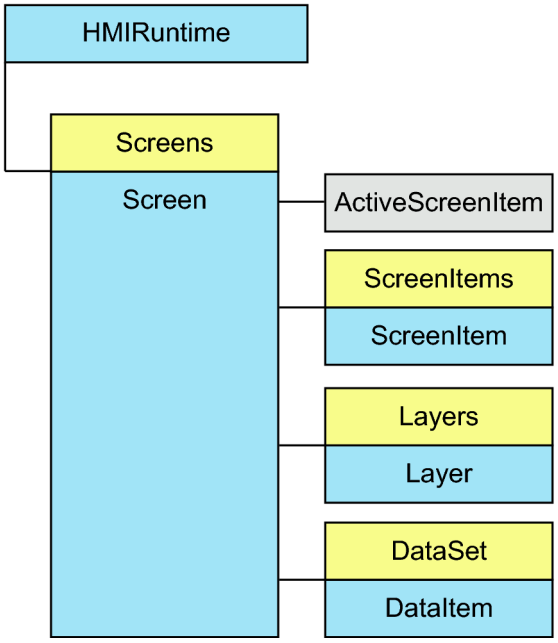
Always enter picture names without the extension "PDL" for reasons of compatibility with future versions.

See also

ScreenItems Property (Page 501)
Refresh Method (Page 702)
Activate Method (Page 639)
Zoom Property (Page 637)
Width Property (Page 626)
Parent Property (Page 471)
ObjectSizeDeclutteringMin Property (Page 457)
ObjectSizeDeclutteringMax Property (Page 457)
ObjectSizeDeclutteringEnable Property (Page 456)
ObjectName Property (Page 455)
Layers Property (Page 423)
DataSet Property (Page 351)
LayerDeclutteringEnable Property (Page 422)
Height Property (Page 391)
FillStyle Property (Page 369)
FillColor Property (Page 368)
ExtendedZoomingEnable Property (Page 367)
Enabled Property (Page 360)
BackColor Property (Page 295)
ActiveScreenItem Property (Page 279)
AccessPath Property (Page 276)

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 134)

Refresh Method (Page 702)

Item Method (Page 687)

Count Property (Page 346)

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.

```
'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.

```
'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:

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

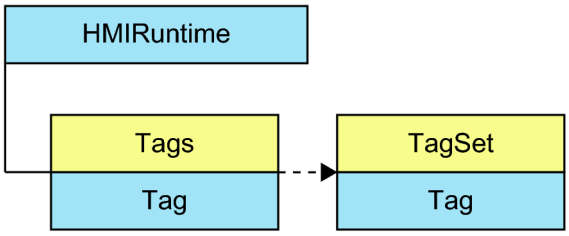
When you activate Runtime, the instance-specific property "wide" is increased by 50 every time you click the button. This increases the rectangle width by 50.

See also

SmartTag property (Page 517)

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_EMPTY). 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

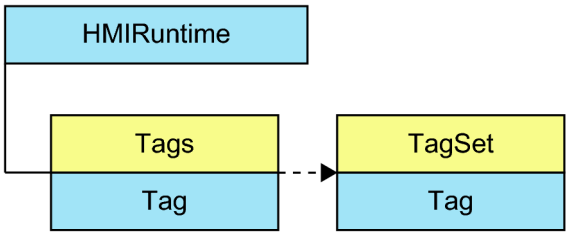
It is not possible to guarantee the compilation of combined tag names from the CrossReference.

See also

Name Property (Page 452)
Example: How to Read Tag Values (Page 741)
Example: Writing tag values (Page 739)
Write Method (Page 725)
Read Method (Page 698)
Value Property (Page 610)
TimeStamp Property (Page 556)
QualityCode Property (Page 487)
LastError Property (Page 407)
ErrorDescription Property (Page 363)

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. In order that the read and write access can be executed without any errors, the corresponding tags must be available in the WinCC Tag Management.

Using VBScript in WinCC it is possible to address tags directly by their name to define and read values. If you wish to inquire about additional tag properties, such as quality code or time stamp, or wish to execute error processing, you must address the tag through tags list. The tag object returned enables access to all tag properties and methods.

Using the "CreateTagSet" method, you may generate a TagSet object which allows 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 741)

Example: Writing tag values (Page 739)

Item Method (Page 687)

CreateTagSet Method (Page 644)

Tag Object (Page 146)

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:

```
'VBS169
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.

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"
'Set 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 407)

Example: How to Read Tag Values (Page 741)

Example: Writing tag values (Page 739)

Write Method (Page 725)

RemoveAll Method (Page 707)

Remove Method (Page 703)

Read Method (Page 698)

Item Method (Page 687)

ErrorDescription Property (Page 363)

Count Property (Page 346)

Add Method (Page 641)

Tags Object (List) (Page 149)

Tag Object (Page 146)

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 138)

ScreenItem Object (Page 134)

Group (Page 274)

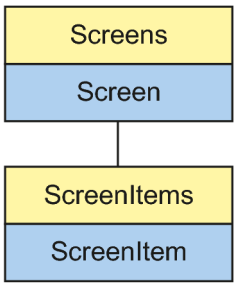
Customized object (Page 273)

Controls (Page 230)

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:

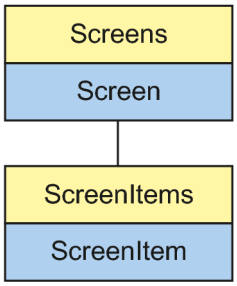
```
'VBS17
Dim objEllipse
Set objEllipse = ScreenItems("Ellipse1")
objEllipse.Left = objEllipse.Left + 10
```

See also

FillStyle Property (Page 369)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
RadiusWidth Property (Page 488)
RadiusHeight Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

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:

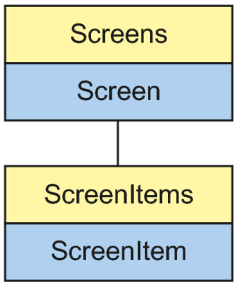
```
'VBS18
Dim objEllipseArc
Set objEllipseArc = ScreenItems("EllipseArc1")
objEllipseArc.Left = objEllipseArc.Left + 10
```

See also

RadiusHeight Property (Page 488)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
StartAngle Property (Page 523)
RadiusWidth Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashBorderColor Property (Page 373)
EndAngle Property (Page 361)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
Layer Property (Page 408)

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:

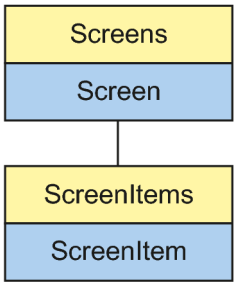
```
'VBS19
Dim objEllipseSeg
Set objEllipseSeg = ScreenItems("EllipseSegment1")
objEllipseSeg.Left = objEllipseSeg.Left + 10
```

See also

Layer Object (Page 127)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
StartAngle Property (Page 523)
RadiusWidth Property (Page 488)
RadiusHeight Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
EndAngle Property (Page 361)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

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:

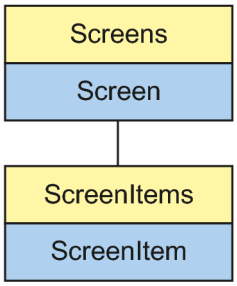
```
'VBS20
Dim objCircle
Set objCircle= ScreenItems("Circle1")
objCircle.Left = objCircle.Left + 10
```

See also

Properties (Page 276)
BorderStyle Property (Page 314)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Radius Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

Circular arc

Description



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

Type Identifier in VBS

HMICircularArc

Usage

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

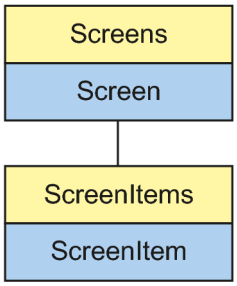
```
'VBS21
Dim objCircularArc
Set objCircularArc = ScreenItems("CircularArc1")
objCircularArc.Left = objCircularArc.Left + 10
```

See also

StartAngle Property (Page 523)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Radius Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashBorderColor Property (Page 373)
EndAngle Property (Page 361)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
Layer Property (Page 408)

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:

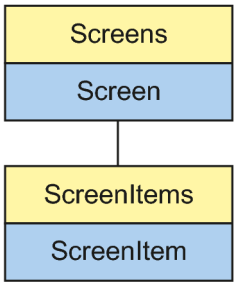
```
'VBS22
Dim objCircleSeg
Set objCircleSeg = ScreenItems("PieSegment1")
objCircleSeg.Left = objCircleSeg.Left + 10
```

See also

Type Property (Page 596)
BorderColor Property (Page 313)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Top Property (Page 575)
ToolTipText Property (Page 574)
StartAngle Property (Page 523)
Radius Property (Page 488)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
EndAngle Property (Page 361)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

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:

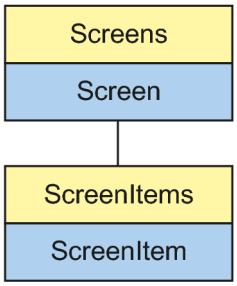
```
'VBS23
Dim objLine
Set objLine = ScreenItems("Line1")
objLine.Left = objLine.Left + 10
```

See also

PasswordLevel Property (Page 472)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
RotationAngle Property (Page 493)
ReferenceRotationTop Property (Page 490)
ReferenceRotationLeft Property (Page 490)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashBorderColor Property (Page 373)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderEndStyle Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
Layer Property (Page 408)

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:

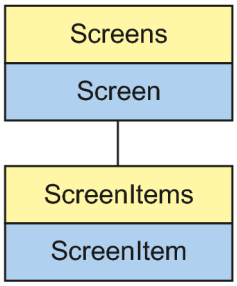
```
'VBS24
Dim objPolygon
Set objPolygon = ScreenItems("Polygon1")
objPolygon.Left = objPolygon.Left + 10
```

See also

ReferenceRotationTop Property (Page 490)
BackFlashColorOn Property (Page 297)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
RotationAngle Property (Page 493)
ReferenceRotationLeft Property (Page 490)
PointCount Property (Page 482)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Index Property (Page 400)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
ActualPointTop Property (Page 280)
ActualPointLeft Property (Page 280)

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:

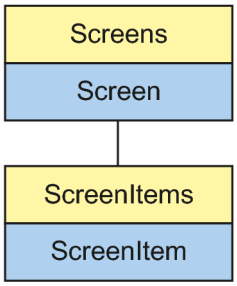
```
'VBS25
Dim objPolyline
Set objPolyline = ScreenItems("Polyline1")
objPolyline.Left = objPolyline.Left + 10
```

See also

Layer Object (Page 127)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
RotationAngle Property (Page 493)
ReferenceRotationTop Property (Page 490)
ReferenceRotationLeft Property (Page 490)
PointCount Property (Page 482)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Index Property (Page 400)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashBorderColor Property (Page 373)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderEndStyle Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
ActualPointTop Property (Page 280)
ActualPointLeft Property (Page 280)
Layer Property (Page 408)

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

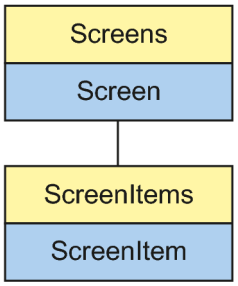
```
Sub OnClick(ByVal Item)
'VBS27
Dim objScreenItem
'
'Activation of errorhandling:
On Error Resume Next
For Each objScreenItem In ScreenItems
If "HMIRectangle" = objScreenItem.Type Then
'
'=== Property "RoundCornerHeight" only available for RoundRectangle
objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 2
If 0 <> Err.Number Then
HMIRuntime.Trace objScreenItem.Name & ": no RoundedRectangle" & vbCrLf
'
'Delete error message
Err.Clear
End If
End If
Next
On Error Goto 0  'Deactivation of errorhandling
End Sub
```

See also

Properties (Page 276)
BorderFlashColorOn Property (Page 314)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

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:

```
'VBS28
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 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

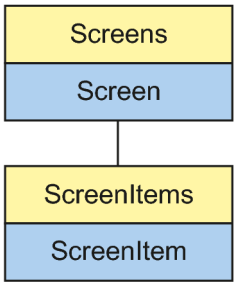
```
Sub OnClick(ByVal Item)
'VBS29
Dim objScreenItem
On Error Resume Next    'Activation of errorhandling
For Each objScreenItem In ScreenItems
If "HMIRectangle" = objScreenItem.Type Then
'
'=== Property "RoundCornerHeight" available only for RoundRectangle
objScreenItem.RoundCornerHeight = objScreenItem.RoundCornerHeight * 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 373)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
RoundCornerWidth Property (Page 494)
RoundCornerHeight Property (Page 493)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)
Layer Property (Page 408)

Static text

Description



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:

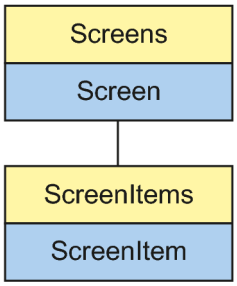
```
'VBS30
Dim objStaticText
Set objStaticText = ScreenItems("StaticText1")
objStaticText.Left = objStaticText.Left + 10
```

See also

ObjectName Property (Page 455)
BorderFlashColorOn Property (Page 314)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Text list (Page 210)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
Orientation Property (Page 468)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
ForeColor Property (Page 383)
FontUnderline Property (Page 382)
FontSize Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashForeColor Property (Page 374)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOff Property (Page 314)

Connector

Description



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
```

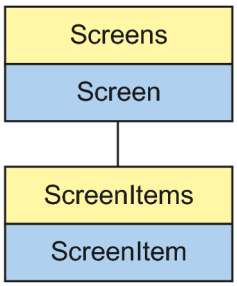
See also

ScreenItems Object (List) (Page 138)
Activate Method (Page 639)
Properties (Page 276)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
TopConnectedObjectName Property (Page 576)
TopConnectedConnectionPointIndex Property (Page 576)
Top Property (Page 575)
ToolTipText Property (Page 574)
Parent Property (Page 471)
Orientation Property (Page 468)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Property (Page 408)
Height Property (Page 391)
Enabled Property (Page 360)
BottomConnectedObjectName Property (Page 316)
BottomConnectedConnectionPointIndex Property (Page 315)

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
```

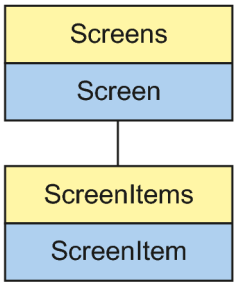
```
On Error Goto 0 'Deactivation of errorhandling
End Sub
```

See also

Type Property (Page 596)
Layer08Color Property (Page 415)
BorderStyle Property (Page 314)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
ZeroPointValue Property (Page 637)
Width Property (Page 626)
Visible Property (Page 624)
Top Property (Page 575)
ToolTipText Property (Page 574)
Process Property (Page 485)
PredefinedAngles Property (Page 484)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Min Property (Page 449)
Max Property (Page 435)
LightEffect Property (Page 424)
Left Property (Page 423)
Layer10Value Property (Page 422)
Layer09Value Property (Page 422)
Layer08Value Property (Page 421)
Layer07Value Property (Page 421)
Layer06Value Property (Page 421)
Layer05Value Property (Page 420)
Layer04Value Property (Page 420)
Layer03Value Property (Page 420)
Layer02Value Property (Page 420)
Layer01Value Property (Page 419)
Layer00Value Property (Page 419)
Layer10Color Property (Page 415)
Layer09Color Property (Page 415)
Layer07Color Property (Page 414)
Layer06Color Property (Page 414)
Layer05Color Property (Page 414)
Layer04Color Property (Page 413)
Layer03Color Property (Page 413)
Layer02Color Property (Page 413)

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:

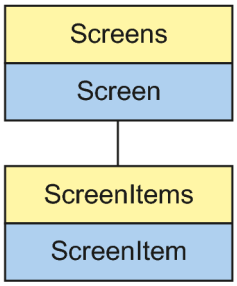
```
'VBS33
Dim objAppWindow
Set objAppWindow = ScreenItems("ApplicationWindow1")
objAppWindow.Left = objAppWindow.Left + 10
```

See also

Properties (Page 276)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
WindowBorder Property (Page 627)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
Template Property (Page 532)
Parent Property (Page 471)
OnTop Property (Page 460)
ObjectName Property (Page 455)
Moveable Property (Page 451)
MaximizeButton Property (Page 435)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
Enabled Property (Page 360)
CloseButton Property (Page 328)
Caption Property (Page 320)
Application Property (Page 287)

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 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

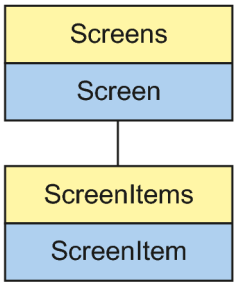
```
'VBS147
Sub OnClick(ByVal Item)
Dim objScreenItem
'
'Activation of errorhandling:
On Error Resume Next
For Each objScreenItem In ScreenItems
If "HMIBar" = objScreenItem.Type Then
'
'=== Property "LimitHigh4" only available for bar
objScreenItem.LimitHigh4 = objScreenItem.LimitHigh4 * 2
If 0 <> Err.Number Then
HMIRuntime.Trace objScreenItem.Name & ": no bar" & vbCrLf
'
'Delete error message
Err.Clear
End If
End If
Next
On Error Goto 0    'Deactivation of errorhandling
End Sub
```

See also

ToolTipText Property (Page 574)
Layer Object (Page 127)
ColorChangeType Property (Page 332)
Average Property (Page 294)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
ZeroPointValue Property (Page 637)
ZeroPoint Property (Page 636)
Width Property (Page 626)
WarningLow Property (Page 626)
WarningHigh Property (Page 626)
Visible Property (Page 624)
TypeWarningLow Property (Page 601)
TypeWarningHigh Property (Page 600)
TypeToleranceLow Property (Page 600)
TypeToleranceHigh Property (Page 600)
TypeLimitLow5 Property (Page 600)
TypeLimitLow4 Property (Page 599)
TypeLimitHigh5 Property (Page 599)
TypeLimitHigh4 Property (Page 599)
TypeAlarmLow Property (Page 598)
TypeAlarmHigh Property (Page 598)
Type Property (Page 596)
TrendColor Property (Page 579)
Trend Property (Page 577)
Top Property (Page 575)
ToleranceLow Property (Page 562)
ToleranceHigh Property (Page 562)
ScalingType Property (Page 499)
Scaling Property (Page 498)
ScaleTicks Property (Page 498)
ScaleColor Property (Page 498)
RightComma Property (Page 492)
Process Property (Page 485)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Min Property (Page 449)

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:

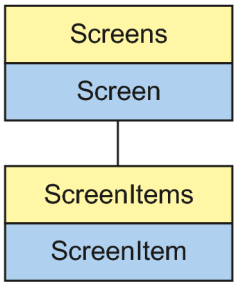
```
'VBS35
Dim objScrWindow
Set objScrWindow = ScreenItems("ScreenWindow1")
objScrWindow.Left = objScrWindow.Left + 10
```

See also

ServerPrefix Property (Page 509)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Zoom Property (Page 637)
WindowBorder Property (Page 627)
Width Property (Page 626)
Visible Property (Page 624)
UpdateCycle Property (Page 603)
Type Property (Page 596)
Top Property (Page 575)
TagPrefix Property (Page 531)
ScrollPositionY Property (Page 502)
ScrollPositionX Property (Page 502)
ScrollBars Property (Page 502)
ScreenName Property (Page 500)
Screens Property (Page 501)
Parent Property (Page 471)
OnTop Property (Page 460)
OffsetTop Property (Page 458)
OffsetLeft Property (Page 458)
ObjectName Property (Page 455)
Moveable Property (Page 451)
MaximizeButton Property (Page 435)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
Enabled Property (Page 360)
CloseButton Property (Page 328)
CaptionText Property (Page 321)
Caption Property (Page 320)
AdaptSize Property (Page 281)
AdaptPicture Property (Page 281)
MenuToolBarConfig Property (Page 442)

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("Control1")
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("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 UserFriendlyName as follows:

```
'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:

```
Control.object.type
```

The properties of the ScreenItem object are used in the case of identical names, if you use the following form:

```
Control.type
```

WinCC controls available

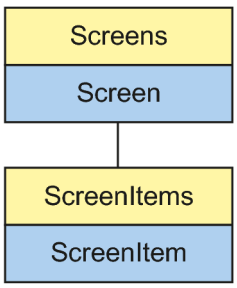
- WinCC Alarm Control
- WinCC Digital/Analog Clock
- WinCC FunctionTrendControl
- WinCC Gauge Control
- WinCC Media Control
- WinCC OnlineTableControl
- WinCC OnlineTrendControl
- WinCC Push Button Control
- WinCC Slider Control
- WinCC UserArchiveControl
- HMI Symbol Library 1.4.1

See also

Object Property (Page 455)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Property (Page 408)
Height Property (Page 391)
Enabled Property (Page 360)

I/O Field

Description



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:

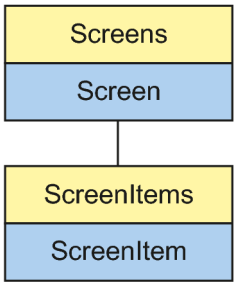
```
'VBS39
Dim objIOField
Set objIOField = ScreenItems("IOField1")
objIOField.Left = objIOField.Left + 10
```

See also

OperationMessage Property (Page 460)
EditAtOnce Property (Page 360)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
OutputValue Property (Page 470)
OutputFormat Property (Page 469)
Orientation Property (Page 468)
OperationReport Property (Page 461)
ObjectName Property (Page 455)
LimitMin Property (Page 426)
LimitMax Property (Page 426)
Left Property (Page 423)
Layer Object (Page 127)
HiddenInput Property (Page 392)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
ForeColor Property (Page 383)
FontUnderline Property (Page 382)
FontSize Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashForeColor Property (Page 374)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillColor Property (Page 368)

Faceplate Instance

Description



Object Type of ScreenItem Object. Represents the "faceplate instance" graphic object.

Type identifier in VBS

HMIFaceplateObject

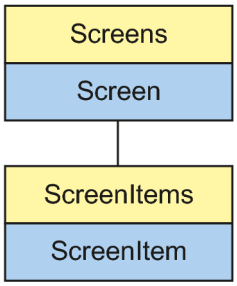
Usage

In the following example, the object with the name "FaceplateInstance1" is moved 10 pixels to the right:

```
'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:

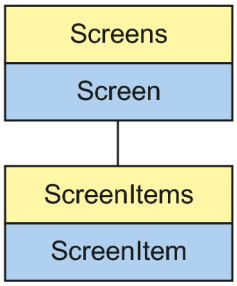
```
'VBS40
Dim objGraphicView
Set objGraphicView= ScreenItems("GraphicObject1")
objGraphicView.Left = objGraphicView.Left + 10
```

See also

Parent Property (Page 471)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
PicUseTransColor Property (Page 482)
PictureName Property (Page 480)
PicTransColor Property (Page 478)
PicReferenced Property (Page 477)
PasswordLevel Property (Page 472)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BackFlashColorOn Property (Page 297)
BackFlashColorOff Property (Page 297)
BackColor Property (Page 295)

Combobox

Description



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

Type Identifier in VBS

HMIComboBox

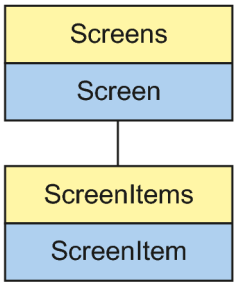
Usage

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

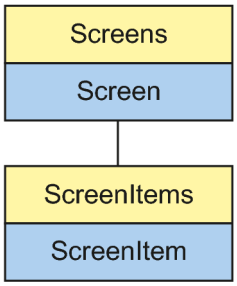
Usage

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

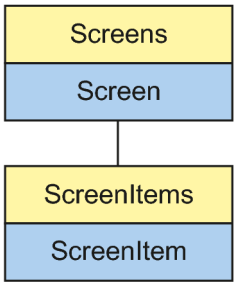
Usage

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
```

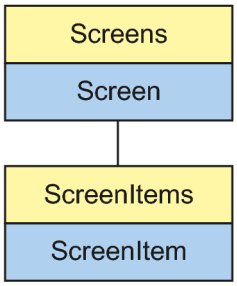
In the case of identical names, the properties of the ScreenItem object are used.

See also

Height Property (Page 391)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Parent Property (Page 471)
Object Property (Page 455)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Property (Page 408)
Enabled Property (Page 360)

Group Display

Description



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:

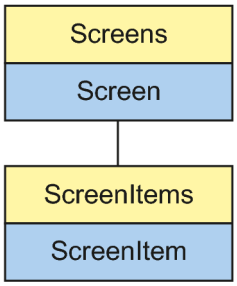
```
'VBS44
Dim objGroupDisplay
Set objGroupDisplay = ScreenItems("GroupDisplay1")
objGroupDisplay.Left = objGroupDisplay.Left + 10
```

See also

Activate Method (Page 639)
MCKQBackColorOn Property (Page 439)
FontBold Property (Page 380)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
UserValue4 Property (Page 608)
UserValue3 Property (Page 607)
UserValue2-Eigenschaft (Page 607)
UserValue1 Property (Page 607)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
SignificantMask Property (Page 516)
SameSize Property (Page 497)
Relevant Property (Page 491)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
MessageClass Property (Page 447)
MCText Property (Page 441)
MCKQTextFlash Property (Page 441)
MCKQTextColorOn Property (Page 440)
MCKQTextColorOff Property (Page 440)
MCKQBackFlash Property (Page 440)
MCKQBackColorOff Property (Page 439)
MCKOTextFlash Property (Page 439)
MCKOTextColorOn Property (Page 439)
MCKOTextColorOff Property (Page 438)
MCKOBackFlash Property (Page 438)
MCKOBackColorOn Property (Page 438)
MCKOBackColorOff Property (Page 437)
MCGUTextFlash Property (Page 437)
MCGUTextColorOn Property (Page 437)
MCGUTextColorOff Property (Page 437)
MCGUBackFlash Property (Page 436)
MCGUBackColorOn Property (Page 436)
MCGUBackColorOff-Eigenschaft (Page 436)

Text list

Description



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:

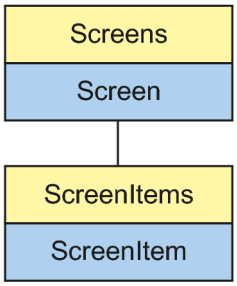
```
'VBS45
Dim objSymIO
Set objSymIO = ScreenItems("TextList1")
objSymIO.Left = objSymIO.Left + 10
```

See also

Type Property (Page 596)
FontUnderline Property (Page 382)
BackFlashColorOff Property (Page 297)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
UnselTextColor Property (Page 603)
UnselBGColor Property (Page 602)
Top Property (Page 575)
ToolTipText Property (Page 574)
SelTextColor Property (Page 508)
SelBGColor Property (Page 504)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
OutputValue Property (Page 470)
Orientation Property (Page 468)
OperationReport Property (Page 461)
OperationMessage Property (Page 460)
ObjectName Property (Page 455)
NumberLines Property (Page 454)
ListType Property (Page 428)
Left Property (Page 423)
Layer Object (Page 127)
LanguageSwitch Property (Page 406)
ItemBorderWidth Property (Page 404)
ItemBorderStyle Property (Page 403)
ItemBorderColor Property (Page 403)
ItemBorderBackColor Property (Page 403)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
ForeColor Property (Page 383)
FontSize Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)

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
```

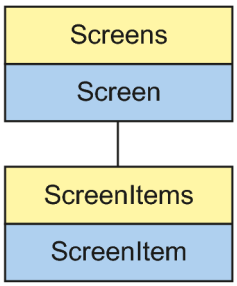
See also

Layer Object (Page 127)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Index Property (Page 400)
Height Property (Page 391)
FlashRateFlashPic Property (Page 376)
FlashRateBorderColor Property (Page 376)
FlashPicUseTransColor Property (Page 375)
FlashPicture Property (Page 375)
FlashPicTransColor Property (Page 374)
FlashPicReferenced Property (Page 374)
FlashFlashPicture Property (Page 373)
FlashBorderColor Property (Page 373)
Enabled Property (Page 360)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)
BorderFlashColorOff Property (Page 314)
BorderColor Property (Page 313)
BorderBackColor Property (Page 312)
BasePicUseTransColor Property (Page 302)
BasePicture Property (Page 302)
BasePicTransColor Property (Page 301)
BasePicReferenced Property (Page 301)

1.14.3.4 Windows objects

Button

Description



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

Type Identifier in VBS

HMIButton

Usage

In the following example, the object with the name "Button1" is moved 10 pixels to the right:

```
'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:

```
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

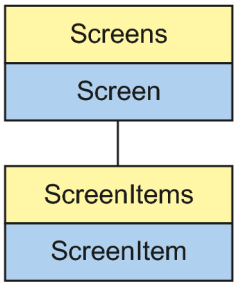
```
Sub OnClick(ByVal Item)
'VBS48
Dim objScreenItem
On Error Resume Next    'Activation of errorhandling
For Each objScreenItem In ScreenItems
If objScreenItem.Type = "HMIButton" Then
'
'=== Property "Text" available only for Standard-Button
objScreenItem.Text = "Windows"
If 0 <> Err.Number Then
HMIRuntime.Trace objScreenItem.ObjectName & ": no Windows-Button" & vbCrLf
Err.Clear    'Delete error message
End If
End If
Next
On Error Goto 0    'Deactivation of errorhandling
End Sub
```

See also

Top Property (Page 575)
FlashBorderColor Property (Page 373)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
WindowsStyle Property (Page 629)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
ToolTipText Property (Page 574)
Text list (Page 210)
PictureUp Property (Page 479)
PictureDown Property (Page 479)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
Orientation Property (Page 468)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Hotkey Property (Page 397)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
ForeColor Property (Page 383)
FontUnderline Property (Page 382)
FontSize Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashForeColor Property (Page 374)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)

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:

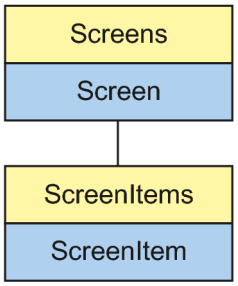
```
'VBS49
Dim chkCheckBox
Set chkCheckBox = ScreenItems("CheckBox1")
chkCheckBox.Left = chkCheckBox.Left + 10
```

See also

FontSize Property (Page 382)
BackColor Property (Page 295)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Text list (Page 210)
Process Property (Page 485)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
Orientation Property (Page 468)
OperationMessage Property (Page 460)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Index Property (Page 400)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
ForeColor Property (Page 383)
FontUnderline Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashForeColor Property (Page 374)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)

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:

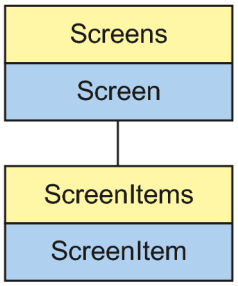
```
'VBS50
Dim objOptionGroup
Set objOptionGroup = ScreenItems("RadioBox1")
objOptionGroup.Left = objOptionGroup.Left + 10
```

See also

ForeColor Property (Page 383)
BackFlashColorOn Property (Page 297)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Text list (Page 210)
Process Property (Page 485)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
Orientation Property (Page 468)
OperationMessage Property (Page 460)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Index Property (Page 400)
Height Property (Page 391)
ForeFlashColorOn Property (Page 384)
ForeFlashColorOff Property (Page 383)
FontUnderline Property (Page 382)
FontSize Property (Page 382)
FontName Property (Page 381)
FontItalic Property (Page 380)
FontBold Property (Page 380)
FlashRateForeColor Property (Page 377)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashForeColor Property (Page 374)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)

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:

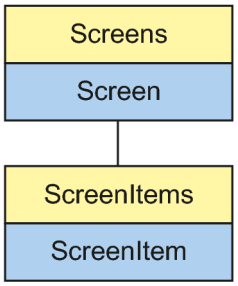
```
'VBS51
Dim objSwitch
Set objSwitch= ScreenItems("RoundButton1")
objSwitch.Left = objSwitch.Left + 10
```

See also

PicDownUseTransColor Property (Page 477)
BorderColorTop Property (Page 313)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Toggle Property (Page 561)
Radius Property (Page 488)
Pressed Property (Page 484)
PicUpUseTransColor Property (Page 481)
PicUpTransparent Property (Page 481)
PicUpReferenced Property (Page 481)
PictureUp Property (Page 479)
PictureDown Property (Page 479)
PictureDeactivated Property (Page 479)
PicDownTransparent Property (Page 477)
PicDownReferenced Property (Page 477)
PicDeactUseTransColor Property (Page 476)
PicDeactTransparent Property (Page 476)
PicDeactReferenced-Eigenschaft (Page 476)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
Enabled Property (Page 360)

Slider

Description



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

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

```
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
```

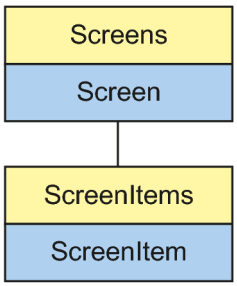
See also

Height Property (Page 391)
BackColorBottom Property (Page 297)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
WindowsStyle Property (Page 629)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
SmallChange Property (Page 517)
Process Property (Page 485)
PasswordLevel Property (Page 472)
Parent Property (Page 471)
OperationReport Property (Page 461)
OperationMessage Property (Page 460)
ObjectName Property (Page 455)
Min Property (Page 449)
Max Property (Page 435)
Left Property (Page 423)
Layer Object (Page 127)
FlashRateBorderColor Property (Page 376)
FlashRateBackColor Property (Page 376)
FlashBorderColor Property (Page 373)
FlashBackColor Property (Page 373)
FillStyle Property (Page 369)
FillingIndex Property (Page 369)
Filling Property (Page 368)
FillColor Property (Page 368)
ExtendedOperation Property (Page 367)
Enabled Property (Page 360)
Direction Property (Page 357)
ColorTop Property (Page 335)
ColorBottom Property (Page 332)
ButtonColor Property (Page 317)
BorderWidth Property (Page 315)
BorderStyle Property (Page 314)
BorderFlashColorOn Property (Page 314)

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

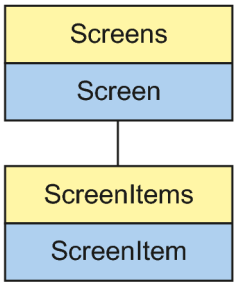
Usage

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

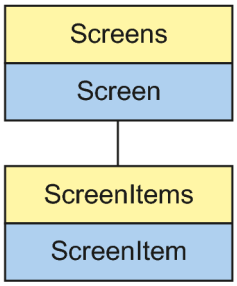
Usage

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

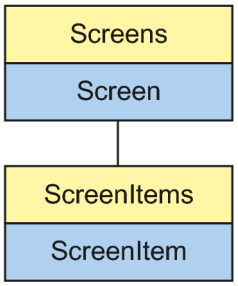
Usage

In the following example, the object with the name "TubeDoubleTeeObject1" is moved 10 pixels to the right:

```
'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:

```
'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:

```
'VBS151
Dim arrayPoints(200)
Dim vArrayCoercion      'Variant for array Coercion
' Make the VBS Array compatibile 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:

```
Control.object.type
```

If you use the following form, the properties of the ScreenItem object are used in the case of identical names:

```
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

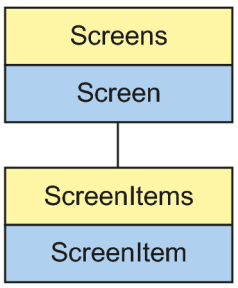
- HMI Symbol Library 1.4.1
- WinCC AlarmControl
- WinCC Alarm Control (before WinCC V7)
- WinCC Digital/Analog Clock
- WinCC FunctionTrendControl
- WinCC Function Trend Control (before WinCC V7)
- WinCC Gauge Control
- WinCC Media Control
- WinCC OnlineTableControl
- WinCC Online Table Control (before 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 1.4.1 (Page 233)
WinCC Slider Control (Page 259)
WinCC Push Button Control (Page 253)
WinCC Online Trend Control (before WinCC V7) (Page 271)
WinCC Online Table Control (before WinCC V7) (Page 269)
WinCC Gauge Control (Page 243)
WinCC Function Trend Control (before WinCC V7) (Page 267)
WinCC Digital/Analog Clock (Page 239)
WinCC Alarm Control (before WinCC V7) (Page 265)
WinCC UserArchiveControl (Page 263)
WinCC RulerControl (Page 257)
WinCC OnlineTrendControl (Page 250)
WinCC OnlineTableControl (Page 247)
WinCC FunctionTrendControl (Page 241)
WinCC AlarmControl (Page 236)

HMI Symbol Library 1.4.1

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

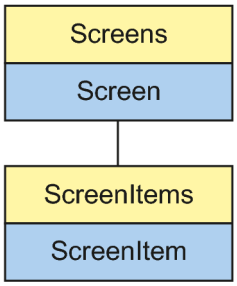
This object type has the following properties:

See also

Left Property (Page 423)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
Stretch Property (Page 529)
Rotation Property (Page 492)
Picture Property (Page 478)
Parent Property (Page 471)
ObjectName Property (Page 455)
Object Property (Page 455)
Layer Object (Page 127)
Height Property (Page 391)
ForeColor Property (Page 383)
Flip Property (Page 377)
Enabled Property (Page 360)
Cursor Property (Page 348)
BlinkSpeed Property (Page 308)
BlinkMode Property (Page 307)
BlinkColor Property (Page 307)
BackStyle Property (Page 299)
BackColor Property (Page 295)

WinCC AlarmControl

Description



Object Type of ScreenItem Object. Represents the graphic object "WinCC AlarmControl" as of WinCC V7.0.

Type Identifier in VBS

HMIAlarmControl

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
```

Available Methods in VBS

Activate	GetOperatorMessage	MoveToNextLine	ShowHitList
ActivateDynamic	GetOperatorMessageCollection	MoveToNextPage	ShowInfoText
AttachDB	GetStatusbarElement	MoveToPreviousLine	ShowLockDialog
CopyRows	GetStatusbarElementCollection	MoveToPreviousPage	ShowLockList

DeactivateDynamic	GetToolbarButton	Print	ShowLongTermArchiveList
DetachDB	GetToolbarButtonCollection	QuitHorn	ShowMessageList
Export	HideAlarm	QuitSelected	ShowPropertyDialog
GetHitlistColumn	LockAlarm	QuitVisible	ShowSelectionDialog
GetHitlistColumnCollection	LoopInAlarm	ShowComment	ShowShortTermArchiveList
GetMessageBlock	MoveToFirstLine	ShowDisplayOptionsDialog	ShowSortDialog
GetMessageBlockCollection	MoveToFirstPage	ShowEmergencyQuitDialog	ShowTimebaseDialog
GetMessageColumn	MoveToLastLine	ShowHelp	UnhideAlarm
GetMessageColumnCollection	MoveToLastPage	ShowHideList	UnlockAlarm

Available Properties in VBS

Activate	HorizontalGridLines	OperatorMessageSourceType3	StatusbarElementWidth
AllServer	IconSpace	OperatorMessageSourceType4	StatusbarShowTooltips
ApplyProjectSettings	LineColor (Page 426)	OperatorMessageSourceType5	StatusbarUseBackColor
AutoCompleteColumns	LineWidth (Page 428)	OperatorMessageSourceType6	StatusbarVisible
AutoCompleteRows	LongTermArchiveConsistency	OperatorMessageSourceType7	TableColor
AutoScroll	MessageBlockAlign	OperatorMessageSourceType8	TableColor2
AutoSelectionColors	MessageBlockAutoPrecisions	OperatorMessageSourceType9	TableForeColor
AutoSelectionRectColor	MessageBlockCaption	PageMode	TableForeColor2
BackColor	MessageBlockDateFormat	PageModeMessageNumber	TimeBase
BorderColor	MessageBlockExponentialFormat	PrintJobName	TitleColor
BorderWidth	MessageBlockFlashOn	RowScrollbar	TitleCut
Caption	MessageBlockHideText	RowTitleAlign	TitleDarkShadowColor
CellCut	MessageBlockHideTitleText	RowTitles	TitleForeColor
CellSpaceBottom	MessageBlockIndex	RTPersistence	TitleGridLineColor
CellSpaceLeft	MessageBlockLeadingZeros	RTPersistencePasswordLevel	TitleLightShadowColor
CellSpaceRight	MessageBlockLength	RTPersistenceType	TitleSort
CellSpaceTop	MessageBlockPrecisions	SelectedCellColor	TitleStyle
Closeable	MessageBlockSelected	SelectedCellForeColor	ToolbarAlignment
ColumnResize	MessageBlockShowDate	SelectedRowColor	ToolbarBackColor
ColumnTitleAlign	MessageBlockShowIcon	SelectedRowForeColor	ToolbarButtonActive

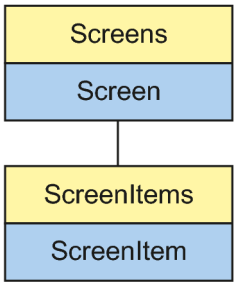
ColumnTitles	MessageBlockShowTitleIcon	SelectedTitleColor	ToolbarButtonAdd
DefaultMsgFilterSQL	MessageBlockTextId	SelectedTitleForeColor	ToolbarButtonBeginGroup
DefaultSort	MessageColumnAdd	SelectionColoring	ToolbarButtonClick
DefaultSort2	MessageColumnIndex	SelectionRect	ToolbarButtonEnabled
DefaultSort2Column	MessageColumnRemove	SelectionRectColor	ToolbarButtonHotKey
DisplayOptions	MessageColumnRepos	SelectionRectWidth	ToolbarButtonId
DoubleClickAction	MessageColumnSort	SelectionType	ToolbarButtonIndex
ExportDirectoryChangeable	MessageColumnSortIndex	ServerNames	ToolbarButtonLocked
ExportDirectoryname	MessageColumnVisible	ShowSortButton	ToolbarButtonName
ExportFileExtension	MessageListType	ShowSortIcon	ToolbarButtonPasswordLevel
ExportFilename	Moveable	ShowSortIndex	ToolbarButtonRemove
ExportFilenameChangeable	MsgFilterSQL	ShowTitle	ToolbarButtonRename
ExportParameters	OperatorMessageNumber	Sizeable	ToolbarButtonRepos
ExportSelection	OperatorMessageSelected	SkinName	ToolbarButtonTooltipText
ExportShowDialog	OperatorMessageSource1	SortSequence	ToolbarButtonVisible
Font	OperatorMessageSource10	StatusbarBackColor	ToolbarShowTooltips
GridLineColor	OperatorMessageSource2	StatusbarElementAdd	ToolbarUseBackColor
GridLineWidth	OperatorMessageSource3	StatusbarElementAutoSize	ToolbarUseHotKeys
HitlistColumnAdd	OperatorMessageSource4	StatusbarElementIconId	ToolbarVisible
HitlistColumnIndex	OperatorMessageSource5	StatusbarElementId	UseMessageColor
HitlistColumnRemove	OperatorMessageSource6	StatusbarElementIndex	UseSelectedTitleColor
HitlistColumnRepos	OperatorMessageSource7	StatusbarElementName	UseTableColor2
HitlistColumnSort	OperatorMessageSource8	StatusbarElementRemove	VerticalGridLines
HitlistColumnSortIndex	OperatorMessageSource9	StatusbarElementRename	
HitlistColumnVisible	OperatorMessageSourceType1	StatusbarElementRepos	
HitListDefaultSort	OperatorMessageSourceType10	StatusbarElementText	
HitListMaxSourceItems	OperatorMessageSourceType2	StatusbarElementTooltipText	
HitListMaxSourceItemsWarn		StatusbarElementVisible	
HitListRelTime			
HitListRelTimeFactor			
HitListRelTimeFactorType			

See also

Controls (Page 230)

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:

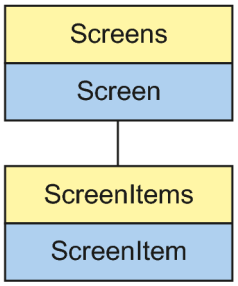
```
'VBS55
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +11
```

See also

Parent Property (Page 471)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
Ticks Property (Page 536)
TicksColor Property (Page 536)
SquareExtent Property (Page 523)
SecondNeedleWidth Property (Page 503)
SecondNeedleHeight Property (Page 503)
Picture Property (Page 478)
ObjectName Property (Page 455)
Object Property (Page 455)
MinuteNeedleWidth Property (Page 450)
MinuteNeedleHeight Property (Page 450)
LocaleID Property (Page 429)
Left Property (Page 423)
Layer Object (Page 127)
HourNeedleWidth Property (Page 398)
HourNeedleHeight Property (Page 398)
Height Property (Page 391)
Handtype Property (Page 391)
HandFillColor Property (Page 390)
ForeColor Property (Page 383)
Font Property (Page 379)
FocusRect Property (Page 378)
Enabled Property (Page 360)
BackStyle Property (Page 299)
BackColor Property (Page 295)
Analog Property (Page 286)

WinCC FunctionTrendControl

Description



Object Type of ScreenItem Object. Represents the graphic object "WinCC FunctionTrendControl" as of WinCC V7.0.

Type Identifier in VBS

HMIFunctionTrendControl

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
```

Available Methods in VBS

Activate	GetToolbarButtonCollection	MoveAxis	ShowTrendSelection
ActivateDynamic	GetTrend	NextTrend	StartStopUpdate
AttachDB	GetTrendCollection	OneToOneView	ZoomArea

DeactivateDynamic	GetTrendWindow	PreviousTrend	ZoomInOut
DetachDB	GetTrendWindowCollection	Print	ZoomInOutX
Export	GetXAxis	ShowHelp	ZoomInOutY
GetStatusbarElement	GetXAxisCollection	ShowPropertyDialog	ZoomMove
GetStatusbarElementCollection	GetYAxis	ShowTagSelection	
GetToolbarButton	GetYAxisCollection	ShowTimeSelection	

Available Properties in VBS

BackColor	StatusbarElementVisible	TrendLowerLimit	TrendXAxis
BorderColor	StatusbarElementWidth	TrendLowerLimitColor	TrendYAxis
BorderWidth	StatusbarShowTooltips	TrendLowerLimitColoring	UseTrendNameAsLabel
Caption	StatusbarUseBackColor	TrendMeasurePoints	XAxisAdd
Closeable	StatusbarVisible	TrendName	XAxisAlign
ConnectTrendWindows	TimeBase	TrendPointColor	XAxisAutoPrecisions
ExportDirectoryChangeable	ToolbarAlignment	TrendPointStyle	XAxisAutoRange
ExportDirectoryname	ToolbarBackColor	TrendPointWidth	XAxisBeginValue
ExportFileExtension	ToolbarButtonActive	TrendProvider	XAxisColor
ExportFilename	ToolbarButtonAdd	TrendProviderCLSID	XAxisEndValue
ExportFilenameChangeable	ToolbarButtonBeginGroup	TrendRangeType	XAxisExponentialFormat
ExportSelection	ToolbarButtonClick	TrendRemove	XAxisIndex
ExportShowDialog	ToolbarButtonEnabled	TrendRename	XAxisInTrendColor
ExportParameters	ToolbarButtonHotKey	TrendRepos	XAxisLabel
Font	ToolbarButtonId	TrendTagNameX	XAxisName
GraphDirection	ToolbarButtonIndex	TrendTagNameY	XAxisPrecisions
LineColor	ToolbarButtonLocked	TrendTimeRangeBase	XAxisRemove
LineWidth	ToolbarButtonName	TrendTimeRangeFactor	XAxisRename
LoadDataImmediately	ToolbarButtonPasswordLevel	TrendTrendWindow	XAxisRepos
Moveable	ToolbarButtonRemove	TrendUncertainColor	XAxisScalingType
Online	ToolbarButtonRename	TrendUncertainColoring	XAxisTrendWindow
PrintJobName	ToolbarButtonRepos	TrendUpperLimit	XAxisVisible
RTPersistence	ToolbarButtonTooltipText	TrendUpperLimitColor	YAxisAdd
RTPersistencePasswordLevel	ToolbarButtonVisible	TrendUpperLimitColoring	YAxisAlign
RTPersistenceType	ToolbarShowTooltips	TrendVisible	YAxisAutoPrecisions
ShowRuler	ToolbarUseBackColor	TrendWindowAdd	YAxisAutoRange
ShowRulerInAxis	ToolbarUseHotKeys	TrendWindowCoarseGrid	YAxisBeginValue
ShowScrollbars	ToolbarVisible	TrendWindowCoarseGridColor	YAxisColor
ShowTitle	TrendActualize	TrendWindowFineGrid	YAxisEndValue
Sizeable	TrendAdd	TrendWindowFineGridColor	YAxisExponentialFormat

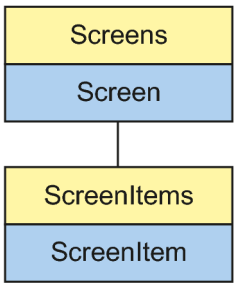
SkinName	TrendBeginTime	TrendWindowForegroundTrendGrid	YAxisIndex
StatusbarBackColor	TrendColor	TrendWindowGridInTrendColor	YAxisInTrendColor
StatusbarElementAdd	TrendEndTime	TrendWindowHorizontalGrid	YAxisLabel
StatusbarElementAutoSize	TrendExtendedColorSet	TrendWindowIndex	YAxisName
StatusbarElementIconId	TrendFill	TrendWindowName	YAxisPrecisions
StatusbarElementId	TrendFillColor	TrendWindowRemove	YAxisRemove
StatusbarElementIndex	TrendIndex	TrendWindowRename	YAxisRepos
StatusbarElementName	TrendLabel	TrendWindowRepos	YAxisScalingType
StatusbarElementRemove	TrendLineStyle	TrendWindowRulerColor	YAxisTrendWindow
StatusbarElementRename	TrendLineType	TrendWindowRulerLayer	YAxisVisible
StatusbarElementRepos	TrendLineWidth	TrendWindowRulerStyle	
StatusbarElementText		TrendWindowRulerWidth	
StatusbarElementTooltipText		TrendWindowSpacePortion	
		TrendWindowVerticalGrid	
		TrendWindowVisible	

See also

Controls (Page 230)

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:

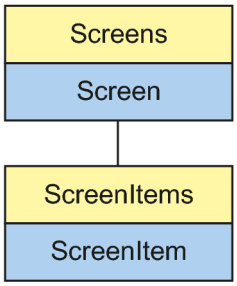
```
'VBS58
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +14
```

See also

WarningColor Property (Page 625)
Object Property (Page 455)
BackColor Property (Page 295)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Object Types of the ScreenItem Object (Page 152)
Width Property (Page 626)
Warning Property (Page 625)
Visible Property (Page 624)
ValueMin Property (Page 624)
ValueMax Property (Page 623)
ValueColumnAlignment Property (Page 617)
UnitText Property (Page 602)
UnitOffset Property (Page 602)
UnitFont Property (Page 601)
UnitColor Property (Page 601)
Type Property (Page 596)
Top Property (Page 575)
TicWidth Property (Page 536)
TicTextOffset Property (Page 535)
TicTextColor Property (Page 535)
TicOffset Property (Page 535)
TicFont Property (Page 534)
TicColor Property (Page 534)
ShowWarning Property (Page 515)
ShowPeak Property (Page 511)
ShowNormal Property (Page 511)
ShowDecimalPoint Property (Page 510)
ShowDanger Property (Page 510)
Rectangular Property (Page 489)
Parent Property (Page 471)
ObjectName Property (Page 455)
NormalColor Property (Page 454)
NeedleColor Property (Page 453)
LocaleID Property (Page 429)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)

WinCC Media Control

Description



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

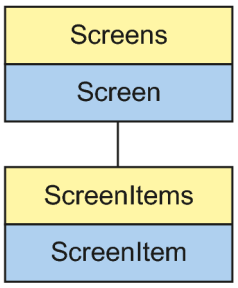
Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'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

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
```

Available Methods in VBS

Activate	Export	GetValueColumnCollection	SelectedStatisticArea
ActivateDynamic	GetStatusbarElement	MoveToFirst	ShowColumnSelection

AttachDB	GetStatusbarElementCollection	MoveToLast	ShowHelp
CalculateStatistic	GetTimeColumn	MoveToNext	ShowPropertyDialog
CopyRows	GetTimeColumnCollection	MoveToPrevious	ShowTagSelection
DeactivateDynamic	GetToolbarButton	NextColumn	ShowTimeSelection
DetachDB	GetToolbarButtonCollection	PreviousColumn	StartStopUpdate
Edit	GetValueColumn	Print	

Available Properties in VBS

AutoCompleteColumns	SelectedRowColor	TimeColumnBeginTime	ToolbarButtonIndex
AutoCompleteRows	SelectedRowForeColor	TimeColumnCaption	ToolbarButtonLocked
AutoSelectionColors	SelectedTitleColor	TimeColumnDateFormat	ToolbarButtonName
AutoSelectionRectColor	SelectedTitleForeColor	TimeColumnEndTime	ToolbarButtonPasswordLevel
BackColor	SelectionColoring	TimeColumnForeColor	ToolbarButtonRemove
BorderColor	SelectionRect	TimeColumnHideText	ToolbarButtonRename
BorderWidth	SelectionRectColor	TimeColumnHideTitleText	ToolbarButtonRepos
Caption	SelectionRectWidth	TimeColumnIndex	ToolbarButtonTooltipText
CellCut	SelectionType	TimeColumnLength	ToolbarButtonVisible
CellSpaceBottom	ShowSortButton	TimeColumnMeasurePoints	ToolbarShowTooltips
CellSpaceLeft	ShowSortIcon	TimeColumnName	ToolbarUseBackColor
CellSpaceRight	ShowSortIndex	TimeColumnRangeType	ToolbarUseHotKeys
CellSpaceTop	ShowTitle	TimeColumnRemove	ToolbarVisible
Closeable	Sizeable	TimeColumnRename	UseColumnBackColor
ColumnResize	SkinName	TimeColumnRepos	UseColumnForeColor
ColumnScrollbar	SortSequence	TimeColumnShowDate	UseSelectedTitleColor

ColumnTitleAlign	StatusbarBackColor	TimeColumnShowIcon	UseTableColor2
ColumnTitles	StatusbarElementAdd	TimeColumnShowTitleIcon	ValueColumnAdd
EnableEdit	StatusbarElementAutoSize	TimeColumnSort	ValueColumnAlign
ExportDirectoryChangeable	StatusbarElementIconId	TimeColumnSortIndex	ValueColumnAutoPrecisions
ExportDirectoryname	StatusbarElementId	TimeColumnTimeFormat	ValueColumnBackColor
ExportFileExtension	StatusbarElementIndex	TimeColumnTimeRangeBase	ValueColumnCaption
ExportFilename	StatusbarElementName	TimeColumnTimeRangeFactor	ValueColumnExponentialFormat
ExportFilenameChangeable	StatusbarElementRemove	TimeColumnUseValueColumnColors	ValueColumnForeColor
ExportParameters	StatusbarElementRename	TimeColumnVisible	ValueColumnHideText
ExportSelection	StatusbarElementRepos	TimeStepBase	ValueColumnHideTitleText
ExportShowDialog	StatusbarElementText	TimeStepFactor	ValueColumnIndex
Font	StatusbarElementTooltipText	TitleColor	ValueColumnLength
GridLineColor	StatusbarElementVisible	TitleCut	ValueColumnName
GridLineWidth	StatusbarElementWidth	TitleDarkShadowColor	ValueColumnPrecisions
HorizontalGridLines	StatusbarShowTooltips	TitleForeColor	ValueColumnProvider
IconSpace	StatusbarUseBackColor	TitleGridLineColor	ValueColumnProviderCLSID
LineColor	StatusbarVisible	TitleLightShadowColor	ValueColumnRemove
LineWidth	TableColor	TitleSort	ValueColumnRename
LoadDataImmediately	TableColor2	TitleStyle	ValueColumnRepos
Moveable	TableForeColor	ToolbarAlignment	ValueColumnShowIcon
Online	TableForeColor2	ToolbarBackColor	ValueColumnShowTitleIcon
PrintJobName	TimeBase	ToolbarButtonActive	ValueColumnSort
RowScrollbar	TimeColumnActualize	ToolbarButtonAdd	ValueColumnSortIndex
RowTitleAlign	TimeColumnAdd	ToolbarButtonBeginGroup	ValueColumnState

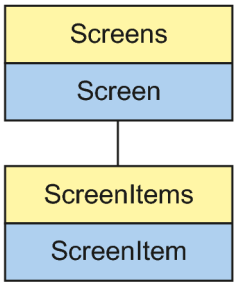
RowTitles	TimeColumnAlign	ToolbarButtonClick	ValueColumnTagName
RTPersistence	TimeColumnBackColor	ToolbarButtonEnabled	ValueColumnTimeColumn
RTPersistencePasswordLevel		ToolbarButtonHotKey	ValueColumnVisible
RTPersistenceType		ToolbarButtonId	VerticalGridLines
SelectedCellColor			
SelectedCellForeColor			

See also

Controls (Page 230)

WinCC OnlineTrendControl

Description



Object Type of ScreenItem Object. Represents the graphic object "WinCC OnlineTrendControl" as of WinCC V7.0.

Type Identifier in VBS

HMIOnlineTrendControl

Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left 16
```

Available Methods in VBS

Activate	GetTimeAxis	MoveToLast	ShowTagSelection
ActivateDynamic method	GetTimeAxisCollection	MoveToNext	ShowTimeSelection
AttachDB method	GetTrend	MoveToPrevious	ShowTrendSelection
CalculateStatistic	GetTrendCollection	NextTrend	StartStopUpdate
DeactivateDynamic	GetTrendWindow	OneToOneView	ZoomArea
DetachDB	GetTrendWindowCollection	PreviousTrend	ZoomInOut
Export	GetValueAxis	Print	ZoomInOutTime
GetRulerData	GetValueAxisCollection	ShowHelp	ZoomInOutValues
GetStatusbarElement	MoveAxis	ShowPercentageAxis	ZoomMove
GetStatusbarElementCollection	MoveToFirst	ShowPropertyDialog	

Available Properties in VBS

BackColor	StatusbarElementRepos	ToolbarButtonRemove	TrendValueAxis
BorderColor	StatusbarElementText	ToolbarButtonRename	TrendVisible
BorderWidth	StatusbarElementTooltipText	ToolbarButtonRepos	TrendWindowAdd
Caption	StatusbarElementVisible	ToolbarButtonTooltipText	TrendWindowCoarseGrid
Closeable	StatusbarElementWidth	ToolbarButtonVisible	TrendWindowCoarseGridColor
ConnectTrendWindows	StatusbarShowTooltips	ToolbarShowTooltips	TrendWindowFineGrid
ExportDirectoryChangeable	StatusbarUseBackColor	ToolbarUseBackColor	TrendWindowFineGridColor
ExportDirectoryname	StatusbarVisible	ToolbarUseHotKeys	TrendWindowForegroundTrendGrid
ExportFileExtension	TimeAxisActualize	ToolbarVisible	TrendWindowGridInTrendColor
ExportFilename	TimeAxisAdd	TrendAdd	TrendWindowHorizontalGrid
ExportFilenameChangeable	TimeAxisAlign	TrendAutoRangeBeginTagName	TrendWindowIndex
ExportParameters	TimeAxisBeginTime	TrendAutoRangeBeginValue	TrendWindowName

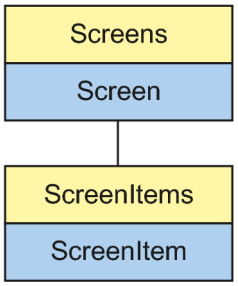
ExportSelection	TimeAxisColor	TrendAutoRangeEndTagName	TrendWindowRemove
ExportShowDialog	TimeAxisDateFormat	TrendAutoRangeEndValue	TrendWindowRename
Font	TimeAxisEndTime	TrendAutoRangeSource	TrendWindowRepos
GraphDirection	TimeAxisIndex	TrendColor	TrendWindowRulerColor
LineColor	TimeAxisInTrendColor	TrendExtendedColorSet	TrendWindowRulerLayer
LineWidth	TimeAxisLabel	TrendFill	TrendWindowRulerStyle
LoadDataImmediately	TimeAxisMeasurePoints	TrendFillColor	TrendWindowRulerWidth
Moveable	TimeAxisName	TrendIndex	TrendWindowSpacePortion
Online	TimeAxisRangeType	TrendLabel	TrendWindowStatisticRulerColor
PercentageAxis	TimeAxisRemove	TrendLineStyle	TrendWindowStatisticRulerStyle
PercentageAxisAlign	TimeAxisRename	TrendLineType	TrendWindowStatisticRulerWidth
PercentageAxisColor	TimeAxisRepos	TrendLineWidth	TrendWindowVerticalGrid
PrintJobName	TimeAxisShowDate	TrendLowerLimit	TrendWindowVisible
RTPersistence	TimeAxisTimeFormat	TrendLowerLimitColor	UseTrendNameAsLabel
RTPersistencePasswordLevel	TimeAxisTimeRangeBase	TrendLowerLimitColoring	ValueAxisAdd
RTPersistenceType	TimeAxisTimeRangeFactor	TrendName	ValueAxisAlign
ShowRuler	TimeAxisTrendWindow	TrendPointColor	ValueAxisAutoPrecisions
ShowRulerInAxis	TimeAxisVisible	TrendPointStyle	ValueAxisAutoRange
ShowScrollbars	TimeBase	TrendPointWidth	ValueAxisBeginValue
ShowStatisticRuler	ToolbarAlignment	TrendProvider	ValueAxisColor
ShowTitle	ToolbarBackColor	TrendProviderCLSID	ValueAxisEndValue
Sizeable	ToolbarButtonActive	TrendRemove	ValueAxisExponentialFormat
SkinName	ToolbarButtonAdd	TrendRename	ValueAxisIndex
StatusbarBackColor	ToolbarButtonBeginGroup	TrendRepos	ValueAxisInTrendColor
StatusbarElementAdd	ToolbarButtonClick	TrendTagName	ValueAxisLabel
StatusbarElementAutoSize	ToolbarButtonEnabled	TrendTimeAxis	ValueAxisName
StatusbarElementIconId	ToolbarButtonHotKey	TrendTrendWindow	ValueAxisPrecisions
StatusbarElementId	ToolbarButtonId	TrendUncertainColor	ValueAxisRemove
StatusbarElementIndex	ToolbarButtonIndex	TrendUncertainColoring	ValueAxisRename
StatusbarElementName	ToolbarButtonLocked	TrendUpperLimit	ValueAxisRepos
StatusbarElementRemove	ToolbarButtonName	TrendUpperLimitColor	ValueAxisScalingType
StatusbarElementRename	ToolbarButtonPasswordLevel	TrendUpperLimitColoring	ValueAxisTrendWindow
			ValueAxisVisible

See also

Controls (Page 230)

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:

```
'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 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

```
'VBS62
Dim objScreenItem
On Error Resume Next    'Activation of errorhandling
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"
```

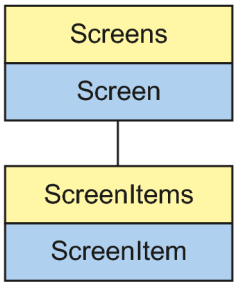
```
If 0 <> Err.Number Then
HMIRuntime.Trace objScreenItem.ObjectName & ": no Control" & vbCrLf
Err.Clear
End If
End If
Next
On Error Goto 0    'Deactivation of errorhandling
```

See also

Properties (Page 276)
FontName Property (Page 381)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Transparent Property (Page 577)
Top Property (Page 575)
PictureUnselected Property (Page 480)
PictureSelected Property (Page 480)
Parent Property (Page 471)
Outline Property (Page 469)
ObjectName Property (Page 455)
Object Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
FrameWidth Property (Page 386)
FrameColorUp Property (Page 385)
FrameColorDown Property (Page 385)
ForeColor Property (Page 383)
FontUnderline Property (Page 382)
FontStrikeThru Property (Page 382)
FontSize Property (Page 382)
FontItalic Property (Page 380)
Font Property (Page 379)
FontBold Property (Page 380)
FocusRect Property (Page 378)
Enabled Property (Page 360)
Caption Property (Page 320)
BackColor Property (Page 295)
AutoSize Property (Page 293)

WinCC RulerControl

Description



Object Type of ScreenItem Object. Represents the graphic object "WinCC RulerControl" as of WinCC V7.0.

Type Identifier in VBS

HMIRulerControl

Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left 16
```

Available Methods in VBS

Activate	GetRulerBlockCollection	GetStatisticResultColumn	GetToolbarButtonCollection
ActivateDynamic	GetRulerColumn	GetStatisticResultColumnCollection	ShowHelp
DeactivateDynamic	GetRulerColumnCollection	GetStatusbarElement	ShowPropertyDialog
Export	GetStatisticAreaColumn	GetStatusbarElementCollection	

GetRulerBlock	GetStatisticAreaColumnCollection	GetToolbarButton

Available Properties in VBS

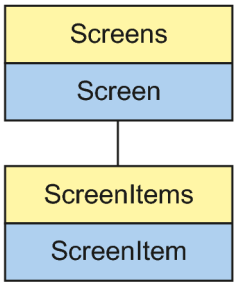
AutoCompleteColumns	ColumnSortIndex	SelectionRectColor	TitleCut
AutoCompleteRows	ColumnTitleAlign	SelectionRectWidth	TitleDarkShadowColor
AutoPositon	ColumnTitles	SelectionType	TitleForeColor
AutoSelectionColors	ColumnVisible	ShowSortButton	TitleGridLineColor
AutoSelectionRectColor	ExportDirectoryChangeable	ShowSortIcon	TitleLightShadowColor
AutoShow	ExportDirectoryname	ShowSortIndex	TitleSort
BackColor	ExportFileExtension	ShowTitle	TitleStyle
BlockAlign	ExportFilename	Sizeable	ToolbarAlignment
BlockAutoPrecisions	ExportFilenameChangeable	SkinName	ToolbarBackColor
BlockCaption	ExportParameters	SortSequence	ToolbarButtonActive
BlockDateFormat	ExportSelection	SourceControl	ToolbarButtonAdd
BlockExponentialFormat	ExportShowDialog	SourceControlType	ToolbarButtonBeginGroup
BlockHideText	Font	StatusbarBackColor	ToolbarButtonClick
BlockHideTitleText	GridLineColor	StatusbarElementAdd	ToolbarButtonEnabled
BlockLength	GridLineWidth	StatusbarElementAutoSize	ToolbarButtonHotKey
BlockPrecisions	HorizontalGridLines	StatusbarElementIconId	ToolbarButtonId
BlockShowDate	IconSpace	StatusbarElementId	ToolbarButtonIndex
BlockShowIcon	LineColor	StatusbarElementIndex	ToolbarButtonLocked
BlockShowTitleIcon	LineWidth	StatusbarElementName	ToolbarButtonName
BlockTimeFormat	Moveable	StatusbarElementRemove	ToolbarButtonPasswordLevel
BlockUseSourceFormat	PrintJobName	StatusbarElementRename	ToolbarButtonRemove
BorderColor	RowScrollbar	StatusbarElementRepos	ToolbarButtonRename
BorderWidth	RowTitleAlign	StatusbarElementText	ToolbarButtonRepos
Caption	RowTitles	StatusbarElementTooltipText	ToolbarButtonTooltipText
CellCut	RTPersistence	StatusbarElementVisible	ToolbarButtonVisible
CellSpaceBottom	RTPersistencePasswordLevel	StatusbarElementWidth	ToolbarShowTooltips
CellSpaceLeft	RTPersistenceType	StatusbarShowTooltips	ToolbarUseBackColor
CellSpaceRight	RulerType	StatusbarUseBackColor	ToolbarUseHotKeys
CellSpaceTop	SelectedCellColor	StatusbarVisible	ToolbarVisible
Closeable	SelectedCellForeColor	TableColor	UseSelectedTitleColor
ColumnAdd	SelectedRowColor	TableColor2	UseSourceBackColors
ColumnIndex	SelectedRowForeColor	TableForeColor	UseSourceForeColors
ColumnRemove	SelectedTitleColor	TableForeColor2	UseTableColor2
ColumnRepos	SelectedTitleForeColor	TitleColor	VerticalGridLines
ColumnResize	SelectionColoring		
ColumnScrollbar	SelectionRect		
ColumnSort			

See also

Controls (Page 230)

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 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)
'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
```

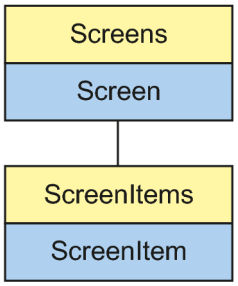
```
End If
End If
Next
On Error GoTo 0   ' deactivating error handling
End Sub
```

See also

PictureThumb Property (Page 480)
BarFillColor Property (Page 300)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
WithLabels Property (Page 630)
WithAxes Property (Page 629)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
TickStyle Property (Page 537)
ThumbBackColor Property (Page 534)
ShowThumb Property (Page 514)
ShowPosition Property (Page 511)
ShowBar Property (Page 510)
RangeMin Property (Page 489)
RangeMax Property (Page 489)
Position Property (Page 482)
PictureBack Property (Page 478)
Parent Property (Page 471)
OuterBevelWidth Property (Page 469)
OuterBevelStyle Property (Page 468)
ObjectName Property (Page 455)
Object Property (Page 455)
LocaleID Property (Page 429)
Left Property (Page 423)
Layer Object (Page 127)
LabelColor Property (Page 405)
InnerBevelWidth Property (Page 401)
InnerBevelStyle Property (Page 401)
InnerBevelOffset Property (Page 401)
Height Property (Page 391)
ForeColor Property (Page 383)
FontPosition Property (Page 381)
Font Property (Page 379)
FocusWidth Property (Page 378)
FocusColor Property (Page 378)

WinCC UserArchiveControl

Description



Object Type of ScreenItem Object. Represents the graphic object "WinCC UserArchiveControl" as of WinCC V7.0.

Type Identifier in VBS

HMIUserArchiveControl

Usage

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

Available Methods in VBS

Activate	GetColumnCollection	MoveToPrevious	ShowSelectArchive
ActivateDynamic	GetStatusbarElement	PasteRows	ShowSelection
CopyRows	GetStatusbarElementCollection	Print	ShowSelectTimeBase
CutRows	GetToolbarButton	ReadTags	ShowSort

DeactivateDynamic	GetToolbarButtonCollection	ServerExport	WriteTags
DeleteRows	MoveToFirst	ServerImport	
Export	MoveToLast	ShowHelp	
GetColumn	MoveToNext	ShowPropertyDialog	

Available Properties in VBS

ArchiveName	ColumnSortIndex	SelectedTitleColor	TimeBase
AutoCompleteColumns	ColumnTimeFormat	SelectedTitleForeColor	TitleColor
AutoCompleteRows	ColumnTitleAlign	SelectionColoring	TitleCut
AutoSelectionColors	ColumnTitles	SelectionRect	TitleDarkShadowColor
AutoSelectionRectColor	ColumnVisible	SelectionRectColor	TitleForeColor
BackColor	EnableDelete	SelectionRectWidth	TitleGridLineColor
BorderColor	EnableEdit	SelectionType	TitleLightShadowColor
BorderWidth	EnableInsert	ShowSortButton	TitleSort
Caption	ExportDirectoryChangeable	ShowSortIcon	TitleStyle
CellCut	ExportDirectoryname	ShowSortIndex	ToolbarAlignment
CellSpaceBottom	ExportFileExtension	ShowTitle	ToolbarBackColor
CellSpaceLeft	ExportFilename	Sizeable	ToolbarButtonActive
CellSpaceRight	ExportFilenameChangeable	SkinName	ToolbarButtonAdd
CellSpaceTop	ExportParameters	SortSequence	ToolbarButtonBeginGroup
Closeable	ExportSelection	StatusbarBackColor	ToolbarButtonClick
ColumnAdd	ExportShowDialog	StatusbarElementAdd	ToolbarButtonEnabled
ColumnAlign	FilterSQL	StatusbarElementAutoSize	ToolbarButtonHotKey
ColumnAutoPrecisions	Font	StatusbarElementIconId	ToolbarButtonId
ColumnCaption	GridLineColor	StatusbarElementId	ToolbarButtonIndex
ColumnDateFormat	GridLineWidth	StatusbarElementIndex	ToolbarButtonLocked
ColumnExponentialFormat	HorizontalGridLines	StatusbarElementName	ToolbarButtonName
ColumnFlagUnique	IconSpace	StatusbarElementRemove	ToolbarButtonPasswordLevel
ColumnHideText	LineColor	StatusbarElementRename	ToolbarButtonRemove
ColumnHideTitleText	LineWidth	StatusbarElementRepos	ToolbarButtonRename
ColumnIndex	Moveable	StatusbarElementText	ToolbarButtonRepos
ColumnLeadingZeros	PrintJobName	StatusbarElementTooltipText	ToolbarButtonTooltipText
ColumnLength	RowScrollbar	StatusbarElementVisible	ToolbarButtonVisible
ColumnPrecisions	RowTitleAlign	StatusbarElementWidth	ToolbarShowTooltips
ColumnReadonly	RowTitles	StatusbarShowTooltips	ToolbarUseBackColor
ColumnRepos	RTPersistence	StatusbarUseBackColor	ToolbarUseHotKeys
ColumnResize	RTPersistencePasswordLevel	StatusbarVisible	ToolbarVisible
ColumnScrollbar	RTPersistenceType	TableColor	UseSelectedTitleColor
ColumnShowDate	SelectedCellColor	TableColor2	UseTableColor2
ColumnShowIcon	SelectedCellForeColor	TableForeColor	VerticalGridLines
ColumnShowTitleIcon	SelectedRowColor	TableForeColor2	

ColumnSort	SelectedRowForeColor

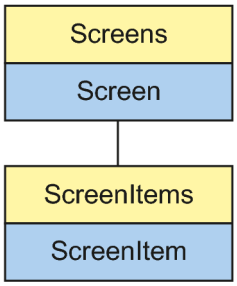
See also

Controls (Page 230)

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:

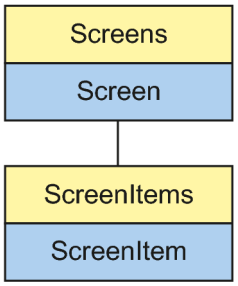
```
'VBS54
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10
```

See also

ProjectPath Property (Page 486)
BackColor Property (Page 295)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
WindowType Property (Page 629)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolbarButtons Property (Page 571)
Titleline Property (Page 560)
TitleCut Property (Page 559)
StatusbarPanes Property (Page 528)
ServerNames Property (Page 509)
SelectionType Property (Page 507)
SelectionRectWidth Property (Page 507)
SelectionRectColor Property (Page 506)
SelectionMode Property (Page 506)
PersistentRTPermission Property (Page 475)
PersistentRTCSPermission Property (Page 475)
Parent Property (Page 471)
ObjectName Property (Page 455)
Object Property (Page 455)
MsgFilterSQL Property (Page 452)
MsgCtrlFlags Property (Page 451)
LineTitle Property (Page 427)
LineHeight Property (Page 427)
LineFont Property (Page 426)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
HeaderSort Property (Page 391)
GridLineVert Property (Page 390)
GridLineHorz Property (Page 388)
Font Property (Page 379)
Enabled Property (Page 360)
ColWidth Property (Page 343)

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:

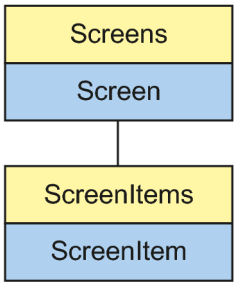
```
'VBS57
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +13
```

See also

Top Property (Page 575)
ScalingTypeY Property (Page 500)
Layer Object (Page 127)
DesiredCurveSourceUAArchive Property (Page 356)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
UpperLimitValue Property (Page 604)
UpperLimit Property (Page 603)
UpperLimitColor Property (Page 604)
Type Property (Page 596)
ToolbarHotKeys Property (Page 572)
ToolbarButtons Property (Page 571)
ToolbarAlignment Property (Page 563)
Titleline Property (Page 560)
TimeZone Property (Page 558)
TimeAxisX Property (Page 544)
TagProviderClsid Property (Page 532)
SourceUAColumnY Property (Page 523)
SourceUAColumnX Property (Page 522)
SourceUAArchiveStartID Property (Page 522)
SourceUAArchive Property (Page 521)
SourceTimeRange Property (Page 521)
SourceTagProviderDataY Property (Page 521)
SourceTagProviderDataX Property (Page 520)
SourceTagNameY Property (Page 520)
SourceTagNameX Property (Page 520)
SourceNumberOfValues Property (Page 519)
SourceNumberOfUAValues Property (Page 519)
SourceEndTime Property (Page 519)
SourceBeginTime Property (Page 518)
ShowValuesExponentialY Property (Page 515)
ShowValuesExponentialX Property (Page 515)
ShowRulerImmediately Property (Page 512)
ScalingTypeX Property (Page 499)
RulerPrecisionY Property (Page 497)

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:

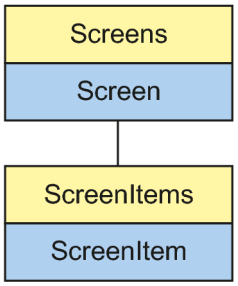
```
'VBS59
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +15
```

See also

TimeOverlap Property (Page 554)
ItemVisible Property (Page 404)
PrintBackgroundColor Property (Page 484)
Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
Variable Property (Page 624)
ValueColumnAlignment Property (Page 617)
UpperLimitValue Property (Page 604)
UpperLimit Property (Page 603)
UpperLimitColor Property (Page 604)
Type Property (Page 596)
Top Property (Page 575)
ToolbarHotKeys Property (Page 572)
Toolbar Property (Page 562)
ToolbarButtons Property (Page 571)
ToolbarAlignment Property (Page 563)
Titleline Property (Page 560)
TimeZone Property (Page 558)
TimeRangeFactor Property (Page 556)
TimeRange Property (Page 555)
TimeRangeBase Property (Page 556)
TimeOverlapColor Property (Page 555)
TimeJump Property (Page 553)
TimeJumpColor Property (Page 554)
TimeFormat Property (Page 553)
TimeColumnAlignment Property (Page 545)
Statusbar Property (Page 524)
PrintJob Property (Page 484)
Precisions Property (Page 483)
PersistentRTPermission Property (Page 475)
PersistentRT Property (Page 474)
PersistentRTCSPermission Property (Page 475)
PersistentRTCS Property (Page 474)
Parent Property (Page 471)
Online Property (Page 459)

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:

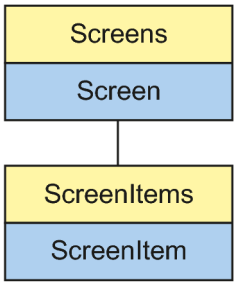
```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left +16
```

See also

Properties (Page 276)
TimeAxis Property (Page 537)
LowerLimitColor Property (Page 434)
Caption Property (Page 320)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Controls (Page 230)
Width Property (Page 626)
Visible Property (Page 624)
UpperLimitValue Property (Page 604)
UpperLimit Property (Page 603)
UpperLimitColor Property (Page 604)
Type Property (Page 596)
Top Property (Page 575)
ToolbarHotKeys Property (Page 572)
Toolbar Property (Page 562)
ToolbarButtons Property (Page 571)
ToolbarAlignment Property (Page 563)
Titleline Property (Page 560)
TimeZone Property (Page 558)
TimeRangeFactor Property (Page 556)
TimeRange Property (Page 555)
TimeRangeBase Property (Page 556)
TimeOverlap Property (Page 554)
TimeOverlapColor Property (Page 555)
TimeJump Property (Page 553)
TimeJumpColor Property (Page 554)
TimeAxisFormat Property (Page 539)
TagName Property (Page 531)
Statusbar Property (Page 524)
ShowRulerImmediately Property (Page 512)
ServerData Property (Page 509)
RulerPrecisions Property (Page 496)
Replacement Property (Page 491)
ReplacementColor Property (Page 491)
RelayCurves Property (Page 490)
ProviderClsid Property (Page 486)
PrintJob Property (Page 484)
Precisions Property (Page 483)

1.14.3.7 Customized object

Description



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

Type Identifier in VBS

HMIScreenModule

Application

Customized properties in a customized object must be accessed in VBS via the attribute name. Intellisense is only applicable to the customized object as a whole.

The attribute name is to be found 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:

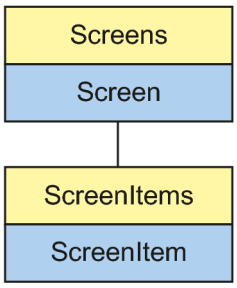
```
'VBS65
Dim objCustomObject
Set objCustomObject = ScreenItems("CustomizedObject1")
objCustomObject.Left = objCustomObject.Left + 10
```

See also

Activate Method (Page 639)
Properties (Page 276)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Object Types of the ScreenItem Object (Page 152)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
Enabled Property (Page 360)

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 276)
Activate Method (Page 639)
ScreenItems Object (List) (Page 138)
ScreenItem Object (Page 134)
Object Types of the ScreenItem Object (Page 152)
Width Property (Page 626)
Visible Property (Page 624)
Type Property (Page 596)
Top Property (Page 575)
ToolTipText Property (Page 574)
Parent Property (Page 471)
ObjectName Property (Page 455)
Left Property (Page 423)
Layer Object (Page 127)
Height Property (Page 391)
Enabled Property (Page 360)

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
```

1.14.4.2 A

Aa - Ad

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 134)

Screens Object (List) (Page 143)

Activate Property

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

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.

The attribute can be assigned dynamic properties by means of the name **Activate** .

ActiveProject Property

Description

Returns an object of type "Project".

See also

Path Property (Page 473)

Name Property (Page 452)

Ellipse segment (Page 158)

HMIRuntime Object (Page 125)

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 focussed object.

Example:

The following example assigns the name of the current picture to the tag "strScrName" and outputs it in a message:

```
'VBS68
Dim strScrName
strScrName = HMIRuntime.ActiveScreen.Objectname
MsgBox strScrName
```

See also

Screen Object (Page 140)

HMIRuntime Object (Page 125)

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:

```
'VBS69
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.ActiveScreenItem.ObjectName
```

See also

ScreenItem Object (Page 134)

HMIRuntime Object (Page 125)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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).

See also

Polyline (Page 170)

Polygon (Page 168)

ScreenItem Object (Page 134)

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 170)

Polygon (Page 168)

ScreenItem Object (Page 134)

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 214)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 191)

ScreenItem Object (Page 134)

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 191)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 116)

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 188)

ScreenItem Object (Page 134)

AlarmLogs Property

Description

Returns an object of type "AlarmLogs".

AlarmLogs (read-only)

See also

HMIRuntime Object (Page 125)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 208)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 208)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

ScreenItem Object (Page 134)

AllowPersistence Property

Description

TRUE, when settings regarding persistence are possible. BOOLEAN write-read access.

See also

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

AllowXAxisColor - Property

Description

TRUE if the defined color of the common X-axis is displayed in runtime. BOOLEAN write-read access.

AllServer Property

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 265)

ScreenItem Object (Page 134)

AllServer property

All servers - AllServer

Selects all servers whose packages were loaded and on which "Alarm Logging Runtime" is activated in the startup list.

Value	Description
TRUE	All servers are activated.
FALSE	Activates only the servers entered in "Server selection".

The attribute can be assigned dynamic properties by means of the name **AllServer**.

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 239)

ScreenItem Object (Page 134)

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 a for the 3D-effect of the "3DBarGraph" object. Value range in degrees from 0 to 90.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 186)

ScreenItem Object (Page 134)

ApplyProjectSettings property

Apply project settings - ApplyProjectSettings

Activates the project settings derived from "Alarm Logging".

Value	Description
TRUE	The "Apply project settings" check box is activated. The message blocks configured in "Alarm Logging" and their properties are activated in AlarmControl. The message blocks are displayed with these properties in the message window.
FALSE	The "Apply project settings" check box is deactivated. You can add or remove message blocks, or edit their properties.

The attribute can be assigned dynamic properties by means of the name **ApplyProjectSettings**.

Ar - Ax

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 269)

ScreenItem Object (Page 134)

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**.

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 210)

ScreenItem Object (Page 134)

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 196)

Text list (Page 210)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

AutoCompleteColumns property

Show blank columns - AutoCompleteColumns

Adds blank columns if the Control width is greater than the width of columns configured.

Value	Description
TRUE	Enables the display of blank columns.
FALSE	Disables the display of blank columns.

The attribute can be assigned dynamic properties by means of the name **AutoCompleteColumns**.

AutoCompleteRows property

Show blank rows - AutoCompleteRows

Enables the insertion of blank rows if the Control length is greater than the number of rows configured.

Value	Description
TRUE	Enables the display of blank rows.
FALSE	Disables the display of blank rows.

The attribute can be assigned dynamic properties by means of the name **AutoCompleteRows**.

AutoPosition property

Automatic positioning - AutoPosition

Defines whether to position the RulerControl exactly below the source control.

The following settings are available:

Value	Description
TRUE	The RulerControl is positioned exactly below the source control.
FALSE	The RulerControl is displayed in accordance with your configuration of the control position.

The attribute can be assigned dynamic properties by means of the name **AutoPosition**.

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

AutoScroll Property

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 265)

ScreenItem Object (Page 134)

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.

Value	Description
TRUE	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.
FALSE	New message events are not selected if "Autoscroll" is disabled. The visible area of the message window is not changed.

The attribute can be assigned dynamic properties by means of the name **AutoScroll**.

AutoSelectionColors property

Automatic selection coloring - AutoSelectionColor

Enables the display of default system colors as selection color for cells and rows.

Value	Description
TRUE	The system colors are in use.
FALSE	The custom colors are used.

The attribute can be assigned dynamic properties by means of the name **AutoSelectionColors**.

AutoSelectionRectColor property

Automatic coloring of selection borders - AutoSelectionRectColor

Defines a system color for the selection border.

Value	Description
TRUE	The system color is in use.
FALSE	The custom color is used.

The attribute can be assigned dynamic properties by means of the name **AutoSelectionRectColors**.

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.

Value	Description
TRUE	The RulerControl is displayed automatically.
FALSE	The RulerControl is not displayed automatically.

The attribute can be assigned dynamic properties by means of the name **AutoShow**.

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 253)

ScreenItem Object (Page 134)

Average Property

Description

TRUE, when the average value from the last 15 values should be calculated. BOOLEAN write-read access.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 134)

Button (Page 214)

Round Button (Page 221)

Slider (Page 223)

Group Display (Page 208)

BackColor property

Background - BackColor

Specifies the background color of the control. Open the "Color selection" dialog by clicking the button.

The attribute can be assigned dynamic properties by means of the name **BackColor**.

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 369)

FillColor Property (Page 368)

ScreenItem Object (Page 134)

BackColor2 Property

Description

Defines or returns the bar color for the display of the current value. LONG write-read access.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

BackColor3 Property

Description

Defines or returns the color of the bar background. LONG write-read access.

See also

ScreenItem Object (Page 134)

Bar (Page 188)

BackColorBottom Property

Description

Defines or returns the color for the bottom/right part of the slider. LONG write-read access.

See also

Slider (Page 223)

ScreenItem Object (Page 134)

BackColorTop Property

Description

Defines or returns the color for the top/left part of the slider. LONG write-read access.

See also

Slider (Page 223)

ScreenItem Object (Page 134)

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 134)

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 134)

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 182)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

BackPictureAlignment property

Description

Defines the mode of representation of the background image in the process picture or the alignment of the picture on the round button, or returns it.

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 1.4.1

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 1.4.1 (Page 233)

WinCC Slider Control (Page 259)

WinCC Gauge Control (Page 243)

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

BarDepth Property

Description

Defines or returns the depth of the bar in pixels.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

BarHeight Property

Description

Defines or returns the height of the bar in pixels.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

BarWidth Property

Description

Defines or returns the width of the bar in pixels.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

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 134)

HMIRuntime Object (Page 125)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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.

See also

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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.

See also

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 134)

WinCC Function Trend Control (before WinCC V7) (Page 267)

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 259)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 210)

ScreenItem Object (Page 134)

BlinkColor Property

Description

Defines the color of the icon in the flash picture. LONG write-read access.

See also

HMI Symbol Library 1.4.1 (Page 233)

ScreenItem Object (Page 134)

BlinkMode Property

Description

Defines the appearance of the icon surface of the flash picture.

- 0: Flashing is switched off.
- 1: The flash picture is invisible.

- 2: The flash picture is assigned a colored, shaded surface. The color of the surface corresponds to the setting of the "BlinkColor" property.
- Solid - 3 (3): The flash picture is assigned a colored, unshaded surface. The color of the surface corresponds to the setting of the "BlinkColor" property.

See also

HMI Symbol Library 1.4.1 (Page 233)

ScreenItem Object (Page 134)

BlinkSpeed Property

Description

Defines the length of the flashing interval.

- Fast - 250: The length of the flashing interval is 250 ms.
- Medium - 500: The length of the flashing interval is 500 ms.
- Slow - 1000: The length of the flashing interval is 1000 ms.

The default value is Medium - 500.

See also

HMI Symbol Library 1.4.1 (Page 233)

ScreenItem Object (Page 134)

BlockAlign property

Block alignment - BlockAlign

Defines the mode of aligning the caption of blocks in column headers.

The following settings are available:

Value	Description	Description
0	left	The block caption is left justified.
1	centered	The block caption is aligned to center.
2	right	The block caption is right justified.

The attribute can be assigned dynamic properties by means of the name **BlockAlign**.

BlockAutoPrecisions property

Decimal places automatic - BlockAutoPrecisions

Enables automatic setting of the decimal precision.

Value	Description
TRUE	The decimal precision is defined automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

The attribute can be assigned dynamic properties by means of the name **BlockAutoPrecisions**.

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**.

BlockDateFormat property

Date format - BlockDateFormat

Defines the date format for visualization.

The following date formats are available:

Value	Description
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.07.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2007.
dd/MM/yy	Day/Month/Year, e.g. 24/12/07.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2007.

The attribute can be assigned dynamic properties by means of the name **BlockDateFormat**.

BlockExponentialFormat property

Exponential notation - BlockExponentialFormat

Specifies exponential notation for the display of values of a selected block.

Value	Description
TRUE	The values are displayed with exponential notation.
FALSE	The values are displayed with decimal notation.

The attribute can be assigned dynamic properties by means of the name **BlockExponentialFormat**.

BlockHideText property

Content as text - BlockHideText

Enables the textual display of the content of a selected block.

Value	Description
TRUE	The content is not displayed in text format. The option is disabled.
FALSE	The content is displayed in text format. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **BlockHideText**.

BlockHideTitleText property

Text header - BlockHideTitleText

Enables the display of the header of a selected block in text format.

Value	Description
TRUE	The header is not displayed in text format. The option is disabled.
FALSE	The header is displayed in text format. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **BlockHideTitleText**.

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**.

BlockPrecisions property

Decimal places - BlockPrecisions

Specifies the decimal precision of values of a 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**.

BlockShowDate property

Display date - BlockShowDate

Specifies that the "Time" block is displayed with the time and date in a field.

Value	Description
TRUE	The date and time are displayed. The date format is defined in the "Date format" field.
FALSE	The time is displayed.

The attribute can be assigned dynamic properties by means of the name **BlockShowDate**.

BlockShowIcon property

Content as icon - BlockShowIcon

Enables the display of the content of a selected block as icon.

Value	Description
TRUE	The content is visualized as icon.
FALSE	The content is not visualized as icon.

The attribute can be assigned dynamic properties by means of the name **BlockShowIcon**.

BlockShowTitleIcon property

Header as icon - BlockShowTitleIcon

Enables the display of the header of a selected block as icon.

Value	Description
TRUE	The header is displayed as icon.
FALSE	The header is not displayed as icon.

The attribute can be assigned dynamic properties by means of the name **BlockShowTitleIcon**.

BlockTimeFormat property

Time format - BlockTimeFormat

Defines the time format to be used for visualization.

The following time formats are available:

Value	Description
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44.240 PM.

The attribute can be assigned dynamic properties by means of the name **BlockTimeFormat**.

BlockUseSourceFormat property

Use source format - BlockUseSourceFormat

Specifies that the format is inherited from the interconnected control.

Value	Description
TRUE	The formats are derived from the interconnected control.
FALSE	The formats entered at this attribute are applied.

The attribute can be assigned dynamic properties by means of the name **BlockUseSouceFormat**.

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 134)

BorderColor Property

Description

Defines or returns the line color for the object. LONG write-read access.

See also

ScreenItem Object (Page 134)

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**.

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 134)

Button (Page 214)

Round Button (Page 221)

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 214)

Round Button (Page 221)

ScreenItem Object (Page 134)

BorderEndStyle Property

Description

Defines or returns the line end style of the object. LONG write-read access.

See also

Polyline (Page 170)

Line (Page 166)

ScreenItem Object (Page 134)

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 134)

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 134)

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 134)

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 134)

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**.

BottomConnectedConnectionPointIndex Property

Description

Specifies or sets the index number of the bottom connecting point.

LONG write-read access.

See also

Connector (Page 180)

ScreenItem Object (Page 134)

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 180)

ScreenItem Object (Page 134)

BoxAlignment Property

Description

TRUE, when the fields are arranged aligned to the right. BOOLEAN write-read access.

See also

Radio box (Page 219)

Check box (Page 217)

ScreenItem Object (Page 134)

BoxCount Property

Description

Defines or returns the number of fields. Value range from 0 to 31.

See also

Radio box (Page 219)

Check box (Page 217)

ScreenItem Object (Page 134)

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 210)

I/O Field (Page 196)

ScreenItem Object (Page 134)

ButtonColor Property

Description

Defines or returns the color of the slider. LONG write-read access.

See also

Slider (Page 223)

ScreenItem Object (Page 134)

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
- 0x00010000; 268435456; Display option dialog

See also

WinCC Alarm Control (before WinCC V7) (Page 265)
ScreenItem Object (Page 134)

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 208)
ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 134)

Group Display (Page 208)

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 208)

ScreenItem Object (Page 134)

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

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 230)

Picture Window (Page 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

Caption property

Text - Caption

Defines the text of the window caption.

The attribute can be assigned dynamic properties by means of the name **Caption**.

CaptionColor Property

Description

Defines or returns the color of the element labeling. LONG write-read access.

See also

ScreenItem Object (Page 134)

WinCC Gauge Control (Page 243)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 191)

ScreenItem Object (Page 134)

CellCut Property

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 265)

ScreenItem Object (Page 134)

CellCut property

Shorten contents - CellCut

Shortens cell contents if the cell width is insufficient.

Value	Description
TRUE	Enables shortening of cell contents.
FALSE	Disables shortening of cell contents.

The attribute can be assigned dynamic properties by means of the name **CellCut**.

CellSpaceBottom property

CellSpaceBottom

Defines the bottom margin of the table cells.

The attribute can be assigned dynamic properties by means of the name **CellSpaceBottom**.

CellSpaceLeft property

CellSpaceLeft

Defines the left indent of the table cells.

The attribute can be assigned dynamic properties by means of the name **CellSpaceLeft** .

CellSpaceRight property

CellSpaceRight

Defines the right indent of the table cells.

The attribute can be assigned dynamic properties by means of the name **CellSpaceRight** .

CellSpaceTop property

CellSpaceTop

Defines the top margin of the table cells.

The attribute can be assigned dynamic properties by means of the name **CellSpaceTop** .

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

Closeable Property

Description

TRUE, when the window can be closed in Runtime. BOOLEAN write-read access.

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

Closeable property

Closeable

Defines whether the control can be closed in Runtime.

Value	Description
TRUE	The control can be closed in Runtime.
FALSE	The control cannot be closed in Runtime.

The attribute can be assigned dynamic properties by means of the name **Closeable**.

CloseButton Property

Description

TRUE, when the window is provided with a "Close" button. Read only access.

See also

Picture Window (Page 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

Co

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 223)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 223)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

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**.

ColumnAlign property

Alignment - ColumnAlign

Specifies the mode of alignment of a selected column.

The following settings are available:

Value	Description	Description
0	left	The selected column is aligned left.
1	Centered	The selected column is aligned to center.
2	right	The selected column is aligned right.

The attribute can be assigned dynamic properties by means of the name **ColumnAlign**.

ColumnAutoPrecisions property

Decimal places automatic - ColumnAutoPrecisions

Enables automatic setting of the decimal precision.

Value	Description
TRUE	The decimal precision is defined automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

The attribute can be assigned dynamic properties by means of the name **ColumnAutoPrecisions**.

ColumnCaption property

Caption - ColumnCaption

Sets the caption for a selected column.

The attribute can be assigned dynamic properties by means of the name **ColumnCaption**.

ColumnDateFormat property

Date format - ColumnDateFormat

Defines the date format for visualization.

The following date formats are available:

Value	Description
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.07.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2007.
dd/MM/yy	Day/Month/Year, e.g. 24/12/07.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2007.

The attribute can be assigned dynamic properties by means of the name **ColumnDateFormat**.

ColumnExponentialFormat property

Exponential notation - ColumnExponentialFormat

Sets exponential notation for the display of values of a selected column.

Value	Description
TRUE	The values are displayed with exponential notation.
FALSE	The values are displayed with decimal notation.

The attribute can be assigned dynamic properties by means of the name **ColumnExponentialFormat**.

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.

Value	Description
TRUE	The column must have a unique value.
FALSE	The column must not have a unique value.

The attribute can be assigned dynamic properties by means of the name **ColumnFlagUnique**.

ColumnHideText property

Content as text - ColumnHideText

Defines textual display of the contents of a selected column.

Value	Description
TRUE	The content is not displayed in text format. The option is disabled.
FALSE	The content is displayed in text format. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **ColumnHideText**.

ColumnHideTitleText property

Text header - ColumnHideTitleText

Sets textual display of the header of a selected column.

Value	Description
TRUE	The header is not displayed in text format. The option is disabled.
FALSE	The header is displayed in text format. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **ColumnHideTitleText**.

ColumnIndex property

ColumnIndex

References a user archive column. Using this attribute you can assign the values of other attributes to a user archive column.

Values between 0 and "ColumnCount" minus 1 are valid for "ColumnIndex". Attribute "ColumnCount" defines the number of available columns.

The "ColumnIndex" attribute can be assigned dynamic properties by means of attribute **ColumnIndex**.

ColumnLeadingZeros property

With leading zeros - ColumnLeadingZeros

Enables the display of values with leading zeros for the column selected.

Value	Description
TRUE	Displays leading zeros. Enter the number of leading zeros in the "Decimal places" field.
FALSE	Disables the display of leading zeros.

The attribute can be assigned dynamic properties by means of the name **ColumnLeadingZeros**.

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**.

ColumnPrecisions property

Decimal places - ColumnPrecisions

Specifies the decimal precision of the values of a 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**.

ColumnReadonly property

Write protected - ColumnReadonly

Sets the write protection of a selected column.

Value	Description
TRUE	This column is write protected.
FALSE	This column is not write protected. You can edit the column values in Runtime by activating the "Change" option in the General" tab.

The attribute can be assigned dynamic properties by means of the name **ColumnReadonly**.

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**.

ColumnRepos property

Up/Down - ColumnRepos

Changes the order of columns. "Up" and "Down" move the time 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**.

ColumnResize property

Width can be resized - ColumnResize

Enables changes to the width of columns.

Value	Description
TRUE	You can change the width of the columns.
FALSE	You cannot change the width of the columns.

The attribute can be assigned dynamic properties by means of the name **ColumnResize**.

ColumnSrollbar property

Column scroll bars - ColumnScrollbar

Enables the display of column scroll bars.

The following settings are available:

Value	Description	Description
0	no	Column scroll bars are not displayed.
1	as required	Column scroll bars are displayed if vertical space requirements of the control are greater than the actually available display area.
2	always	Column scroll bars are always displayed.

The attribute can be assigned dynamic properties by means of the name **ColumnScrollbar**.

ColumnShowDate property

Display date - ColumnShowDate

Specifies that the block "Time" is displayed with the time and date in a field.

Value	Description
TRUE	The date and time are displayed. The date format is defined in the "Date format" field.
FALSE	The time is displayed.

The attribute can be assigned dynamic properties by means of the name **ColumnShowDate**.

ColumnShowIcon property

Content as icon - ColumnShowIcon

Enables the display the contents of a selected column by means of icon.

Value	Description
TRUE	The content is visualized as icon.
FALSE	The content is not visualized as icon.

The attribute can be assigned dynamic properties by means of the name **ColumnShowIcon**.

ColumnShowTitleIcon property

Header as icon - ColumnShowTitleIcon

Specifies the display of the header of a selected column by means of icon.

Value	Description
TRUE	The header is displayed as icon.
FALSE	The header is not displayed as icon.

The attribute can be assigned dynamic properties by means of the name **ColumnShowTitleIcon**.

ColumnSort property

ColumnSort

Defines the sorting order of the user archive column referenced in the "ColumnIndex" attribute.

The following settings are available:

Value	Description	Description
0	no	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

The attribute can be assigned dynamic properties by means of the name **ColumnSort** .

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**.

ColumnTimeFormat property

Time format - ColumnTimeFormat

Defines the time format to be used for visualization.

The following time formats are available:

Value	Description
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44.240 PM.

The attribute can be assigned dynamic properties by means of the name **ColumnTimeFormat**.

ColumnTitleAlign property

Column header alignment - ColumnTitleAlign

Specifies the type of column header alignment.

The following settings are available:

Value	Description	Description
0	left	The column headers are aligned left.
1	centered	The column headers are aligned to center.

Value	Description	Description
2	right	The column headers are aligned right.
3	Justify to table content	The column headers are justified to fit the corresponding column content.

The attribute can be assigned dynamic properties by means of the name **ColumnTitleAlign**.

ColumnTitles property

Display column header - ColumnTitles

Enables the display of the column header.

Value	Description
TRUE	The column header is displayed.
FALSE	The column header is not displayed.

The attribute can be assigned dynamic properties by means of the name **ColumnTitles**.

ColumnType property

Type - ColumnType

Displays the data type set in the user archive for a selected column.

ColumnVisible property

ColumnVisible

Enables the display of a column referenced by means of "ColumnIndex" attribute.

Value	Description
TRUE	The column is displayed.
FALSE	The column is not displayed.

The attribute can be assigned dynamic properties by means of the name **ColumnVisible** .

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 265)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

Comment property

Description

Reads or sets the Alarm object comment.

See also

Alarms object (list) (Page 116)

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 269)

ScreenItem Object (Page 134)

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 134)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ComputerName property

Description

Returns the name of the computer on which the alarm object was triggered.

ComputerName (readonly)

See also

Alarms object (list) (Page 116)

Context property

Description

Reads or sets the alarm object server prefix.

See also

Alarms object (list) (Page 116)

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.

Value	Description
TRUE	All trend windows configured are connected.
FALSE	The trend windows are displayed separately.

The attribute can be assigned dynamic properties by means of the name **ConnectTrendWindows**.

ContinousChange 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 259)

ScreenItem Object (Page 134)

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
HMIRuntime.Trace "Count: " & HMIRuntime.DataSet.Count & vbNewLine
```

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
```

See also

CreateTagSet Method (Page 644)

TagSet Object (List) (Page 151)

ScreenItems Object (List) (Page 138)

Screens Object (List) (Page 143)

Layers Object (Listing) (Page 129)

DataSet-Objekt (List) (Page 122)

ProcessValues Object (List) (Page 132)

Cu

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 125)

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 134)

HMI Symbol Library 1.4.1 (Page 233)

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 210)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

DataIndex Property

Description

Returns the current index of the data of the current trend.

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

DataLogs Property

Description

Returns an object of type "DataLogs".

DataLogs (read-only)

See also

DataLogs Object (Page 121)

HMIRuntime Object (Page 125)

DataSet Property

Description

Returns an object of type "DataSet".

DataSet (read-only)

See also

DataSet-Objekt (List) (Page 122)

HMIRuntime Object (Page 125)

DataX Property

Description

Inserts a single data record and must be set before calling "InsertData".

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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”.

See also

Example: Calling Methods of an ActiveX Control (Page 748)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

DataY Property

Description

Inserts a single data record and must be set before calling "InsertData".

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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**.

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 aer available:

Value	Description	Description
0	Ascending	The list is updated starting at the lowest row.
1	Descending	The list is updated starting at the top row.

The attribute can be assigned dynamic properties by means of the name **DefaultSort**.

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 can be made:

Value	Description	Explanation
0	Ascending	The list is updated starting from the lowest line.
1	Descending	The list is updated starting at the top row.

The attribute can be assigned dynamic properties by means of the name **DefaultSort2**.

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**.

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.

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

Delta Property

Description

Defines or returns the value difference between two main scale graduation marks. Write/Read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 223)

Bar (Page 188)

3D Bar (Page 182)

ScreenItem Object (Page 134)

DisplayOptions property

Show messages - DisplayOptions

Select the messages to be displayed.

The following selection options are available:

Value	Designation
0	All messages
1	Only displayed messages
2	Only hidden messages

The attribute can be assigned dynamic properties by means of the name **DisplayOptions**.

DisplayOptions property

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

Sepcifies the action to be executed in Runtime by double-clicking on a message line.

The following settings are available:

Value	Description	Description
0	none	No action.
1	Loop-in-alarm	Calls the "Loop-in-alarm" function.
2	Open comments dialog	Calls the "Comments dialog" button function.
3	Open Infotext dialog	Calls the "Infotext dialog" button function.
4	Column-dependent	The action is determined by the column in which you double-clicked.

The attribute can be assigned dynamic properties by means of the name **DoubleClickAction**.

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 269)

ScreenItem Object (Page 134)

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 269)

ScreenItem Object (Page 134)

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 210)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 140)

ScreenItem Object (Page 134)

EnableDelete property

Delete - EnableDelete

Enables deletion of data from the user archive in Runtime.

Value	Description
TRUE	You can delete data from the user archive in Runtime.
FALSE	You cannot delete data from the user archive in Runtime.

The attribute can be assigned dynamic properties by means of the name **EnableDelete**.

EnableEdit property

Change - EnableEdit

Enables editing of the data displayed in Runtime.

Value	Description
TRUE	Enables editing of data in Runtime.
FALSE	Disables editing of data in Runtime.

The attribute can be assigned dynamic properties by means of the name **EnableEdit**.

EnableInsert property

Add - EnableInsert

Enables insertion of data in the user archive in Runtime.

Value	Description
TRUE	You can add data to the user archive in Runtime.
FALSE	You cannot add data to the user archive in Runtime.

The attribute can be assigned dynamic properties by means of the name **EnableInsert**.

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 164)

Circular arc (Page 162)

Ellipse segment (Page 158)

Ellipse arc (Page 156)

ScreenItem Object (Page 134)

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.

See also

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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:

Output	Description
" "	OK
"Operation Failed"	Execution error
"Variable not found"	Tag error
"Server down"	Server not available.
"An error occured for one or several tags"	Multi Tag Error (Error in one or several tags)

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 = HMIRuntime.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 = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "ErrorDescription: " & group.ErrorDescription & vbNewLine
```

The ErrorDescription property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "ErrorDescription: " & group("Motor1").ErrorDescription & vbNewLine
```

See also

LastError Property (Page 407)

QualityCode Property (Page 487)

TagSet Object (List) (Page 151)

Tag Object (Page 146)

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 188)

ScreenItem Object (Page 134)

ExportDirectoryChangeable property

Directory can be changed - ExportDirectoryChangeable

Enables changing of the directory for data export in Runtime.

Value	Description
TRUE	The data export directory can be changed in Runtime.
FALSE	The data export directory cannot be changed in Runtime.

The attribute can be assigned dynamic properties by means of the name **ExportDirectoryChangeable**.

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**.

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**.

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**.

ExportFilenameChangeable property

File can be renamed - ExportFilenameChangeable

Enables renaming of the export file in Runtime.

Value	Description
TRUE	The export file can be renamed in Runtime.
FALSE	The export file cannot be renamed in Runtime.

The attribute can be assigned dynamic properties by means of the name **ExportFilenameChangeable**.

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**.

ExportSelection property

Scope of data export - ExportSelection

Specifies the control's Runtime data to be exported.

The following settings are available:

Value	Description	Description
0	all	All Runtime data of the control is exported.
1	Selection	Selected Runtime data of the control is exported.

The attribute can be assigned dynamic properties by means of the name **ExportSelection**.

ExportShowDialog property

Display Dialog - ExportShowDialog

Enables the display of the export dialog in Runtime.

Value	Description
TRUE	The dialog is displayed in Runtime.
FALSE	The dialog is not displayed in Runtime.

The attribute can be assigned dynamic properties by means of the name **ExportShowDialog**.

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 223)

ScreenItem Object (Page 134)

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 140)

1.14.4.7 F

Fi - Fl

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:

```
'VBS73
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
objScreen.FillStyle = 131075
objScreen.FillColor = RGB(0, 0, 255)
```

See also

FillStyle Property (Page 369)

BackColor Property (Page 295)

ScreenItem Object (Page 134)

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 134)

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

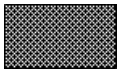
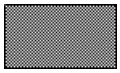
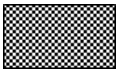
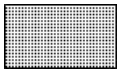
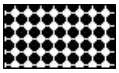
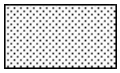
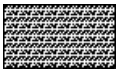
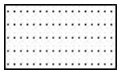
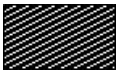
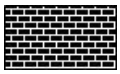
ScreenItem Object (Page 134)

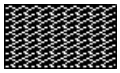
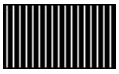
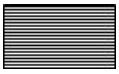
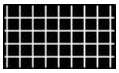
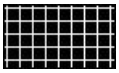
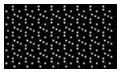
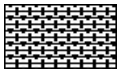
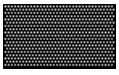
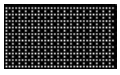
FillStyle Property

Description

Defines or returns the fill pattern for the object.

LONG (write-read access)

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
< Transparent >	65536				
< Massiv >	0				
	1048576		196611		196627
	1048577		196612		196628
	1048578		196613		196629
	1048579		196614		196630
	1048832		196615		196631
	1048833		196616		196632
	1048834		196617		196633

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
	1048835		196618		196634
	131072		196619		196635
	131073		196620		196636
	131074		196621		196637
	131075		196622		196638
	131076		196623		196639
	196608		196624		196640
	196609		196625		196641
	196610		196626		196642

Example:

The following example sets the fill pattern for "ScreenWindow1" to transparent:

```
'VBS190
Dim myScr
Set myScr = HMIRuntime.Screens("PictureWindow1")
myScr.FillStyle = 65536
```

See also

FillColor Property (Page 368)

BackColor Property (Page 295)

Screen Object (Page 140)

FillStyle2 Property

Description

Defines or returns the fill style of the bar.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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**.

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 271)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

FlashBackColor Property

Description

TRUE, when flashing of the background is activated. BOOLEAN write-read access

See also

ScreenItem Object (Page 134)

FlashBorderColor Property

Description

TRUE, when flashing of the object lines is activated. BOOLEAN write-read access.

See also

ScreenItem Object (Page 134)

FlashFlashPicture Property

Description

TRUE, when flashing of the flash picture is activated. BOOLEAN write-read access.

See also

Status display (Page 212)

ScreenItem Object (Page 134)

FlashForeColor Property

Description

TRUE, when flashing of the text is activated. BOOLEAN write-read access.

See also

I/O Field (Page 196)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

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 134)

Status display (Page 212)

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 212)

ScreenItem Object (Page 134)

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 212)

ScreenItem Object (Page 134)

FlashRate Property

Description

Defines or returns the flash frequency. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

FlashRateBackColor Property

Description

Defines or returns the flash frequency for the object background. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

See also

ScreenItem Object (Page 134)

FlashRateBorderColor Property

Description

Defines or returns the flash frequency for the lines of the object. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

See also

ScreenItem Object (Page 134)

FlashRateFlashPic Property

Description

Defines or returns the flash frequency for the flash picture. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

See also

Status display (Page 212)

ScreenItem Object (Page 134)

FlashRateForeColor Property

Description

Defines or returns the flash frequency for the object label. Value range from 0 to 2.

0 = slow

1 = medium

2 = fast

See also

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

ScreenItem Object (Page 134)

Flip Property

Description

Mirrors the icon on the vertical and/or horizontal middle axis of the icon.

- Zero - 0: The icon is mot 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 1.4.1 (Page 233)

ScreenItem Object (Page 134)

Fo - Fr

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 259)

ScreenItem Object (Page 134)

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 253)

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

Font property

Name - Font

Sets the font.

The attribute can be assigned dynamic properties by means of the name **Font**.

Font Property

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 259)

WinCC Push Button Control (Page 253)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

WinCC Digital/Analog Clock (Page 239)

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

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 253)

Group Display (Page 208)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

Bar (Page 188)

ScreenItem Object (Page 134)

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 253)

Group Display (Page 208)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 253)

Group Display (Page 208)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

Bar (Page 188)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

FontSize Property

Description

Defines or returns the font size of the text in the object in points.

See also

WinCC Push Button Control (Page 253)

Group Display (Page 208)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

Bar (Page 188)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 253)
Group Display (Page 208)
Static text (Page 178)
Text list (Page 210)
Radio box (Page 219)
Check box (Page 217)
Button (Page 214)
I/O Field (Page 196)
ScreenItem Object (Page 134)

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 259)
WinCC Push Button Control (Page 253)
WinCC Digital/Analog Clock (Page 239)
HMI Symbol Library 1.4.1 (Page 233)
Static text (Page 178)
Text list (Page 210)
Radio box (Page 219)
Check box (Page 217)
Button (Page 214)
I/O Field (Page 196)
ScreenItem Object (Page 134)

ForeFlashColorOff Property

Description

Defines or returns the color of the text for flash status "Off". LONG write-read access.

See also

Text list (Page 210)
Static text (Page 178)
Radio box (Page 219)
Check box (Page 217)
Button (Page 214)
I/O Field (Page 196)
ScreenItem Object (Page 134)

ForeFlashColorOn Property

Description

Defines or returns the color of the text for flash status "On". LONG write-read access.

See also

Static text (Page 178)
Text list (Page 210)
Radio box (Page 219)
Check box (Page 217)
Button (Page 214)
I/O Field (Page 196)
ScreenItem Object (Page 134)

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 243)
ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

FrameWidth Property

Description

Defines or returns the border width of the button in pixels. Write/Read access.

See also

WinCC Push Button Control (Page 253)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

GraphDirection property

Writer orientation - GraphDirection

Defines the orientation of the update of axis values.

Value	Description	Description
0	From the right	The updated values are displayed starting at the right side of the trend.
1	From the left	The updated values are displayed starting at the left side of the trend.
2	From the top	The updated values are displayed starting at the top of the trend.
3	From the bottom	The updated values are displayed starting at the bottom of the trend.

True type fonts must be used within the trend window if "From the top" or "From the bottom" is selected for writer orientation. 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**.

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**.

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 265)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

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**.

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 239)

ScreenItem Object (Page 134)

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 239)

ScreenItem Object (Page 134)

HeaderSort Property

Description

Specifies if sorting of messages by message block column header is possible.

See also

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

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":

```
'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 626)

Object Types of the ScreenItem Object (Page 152)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

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** .

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**.

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**.

HitlistColumnRepos

Up/Down - Message/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**.

HitlistColumnSort property

HitlistColumnSort

Defines the sorting order of the message block referenced in "HitlistColumnIndex" in the hitlist.

The following settings are available:

Value	Description	Description
0	none	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

The attribute can be assigned dynamic properties by means of the name **HitlistColumnSort** .

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**.

HitlistColumnVisible

Selected message blocks - Message/HitlistColumnVisible

The list shows the message blocks of the message list or hitlist which are selected for use in Runtime Control.

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**.

HitlistDefaultSort property

HitlistDefaultSort

Defines the default sorting order in the table columns of the hitlist.

The following settings can be made:

Value	Description	Explanation
0	Ascending	The list is sorted in ascending order based on frequency.
1	Descending	The list is sorted in descending order based on frequency.

The attribute can be assigned dynamic properties by means of the name **HitlistDefaultSort**.

HitlistMaxSourceItems Property

Description

With this attribute, you can specify the maximum number of records to be used from the message archive for the generation of the hit list. You can choose any value between 0 and 10000.

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** .

HitlistMaxSourceItemsWarn property

Description

With this attribute, you can define whether a warning is issued when the maximum number of records as specified in "HitlistMaxSourceItems" is reached.

HitListMaxSourceItemsWarn property

Warning notice - HitListMaxSourceItemsWarn

Enables the output of a warning notice after the valid number of data records was reached.

Value	Description
TRUE	A warning is output after the valid maximum number of data records was reached.
FALSE	A warning is not output after the valid maximum number of data records was reached.

The attribute can be assigned dynamic properties by means of the name **HitListMaxSourceItemsWarn**.

HitListRelTime property

Time range for statistics - HitListRelTime

Sets a time range for the statistics.

Value	Description
TRUE	The time range set for statistics is used if this range was not defined in the selection.
FALSE	The time range is not used.

The attribute can be assigned dynamic properties by means of the name **HitListRelTime**.

HitlistRelTimeFactor Property

Description

With this attribute, you can define the time factor used in connection with time type "HitlistRelTimeFactorType" to define the time period for which the hit list statistics are generated.

If you do not wish to specify a time period, set the factor to "0".

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**.

HitlistRelTimeFactorType Property

Description

With this attribute, you can define the time type used in connection with time factor "HitlistRelTimeFactor" to define the time period for which the hit list statistics are generated.

Value	Description
0	MSG_TIMEFACTOR_MINUTE: The time period base for the generation of the statistics is "minute".
1	MSG_TIMEFACTOR_HOUR: The time period base for the generation of the statistics is "hour".
2	MSG_TIMEFACTOR_DAY: The time period base for the generation of the statistics is "day".

3	MSG_TIMEFACTOR_WEEK: The time period base for the generation of the statistics is "week".
4	MSG_TIMEFACTOR_MONTH: The time period base for the generation of the statistics is "month".

If you do not wish to define a time period for the statistics set the time factor "HitlistRelTimeFactor" to "0".

HitListRelTimeFactorType property

Time range - HitListRelTimeFactorType

Defines the time unit for calculating the time range.

The following time units are available:

Value	Description
0	Minute
1	Hour
2	Day
3	Week
4	Month

The attribute can be assigned dynamic properties by means of the name **HitListMaxRelTimeFactorType**.

Ho - Hy

HorizontalGridLines property

Horizontal - HorizontalGridLines

Defines whether horizontal separating lines will be displayed.

Value	Description
TRUE	Enables the display of horizontal dividers.
FALSE	Disables the display of horizontal dividers.

The attribute can be assigned dynamic properties by means of the name **HorizontalGridLines**.

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 214)

ScreenItem Object (Page 134)

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 pixels / 2) * 0.5 = 25 pixels.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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 pixels * 2 * 0.1 = 5 pixels.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

Hysteresis Property

Description

TRUE, when the display should appear with hysteresis. BOOLEAN write-read access.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

1.14.4.10 I

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**.

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 change with the shift or scaling of the process picture.

FALSE if the size and position of the picture window are independent of the process picture and only defined by the "Window mode" attribute.

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 212)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

Polyline (Page 170)

Polygon (Page 168)

Radio box (Page 219)

Check box (Page 217)

ScreenItem Object (Page 134)

InnerBevelOffset Property

Description

Defines the distance between the inner and outer bevels.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

InnerBevelWidth Property

Description

Defines the width of the inner bevel in pixels.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

InputValue property

Description

Defines or returns the value specified by the user in the IO field.

You can configure the IO field with this attribute so that the input value changed by the user is also displayed in the IO field after confirming with the <Return> key.

This requires that you configure a direct connection between the "Input value" and "Output value" attributes. Without this direct connection, the entered value is deleted after acceptance with the <Return> key. 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.

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".

See also

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

Instance property

Description

Returns an instance of the alarm object.

See also

Alarms object (list) (Page 116)

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 210)

ScreenItem Object (Page 134)

ItemBorderColor Property

Description

Defines or returns the color for deviding lines in the selection list of the text list object. LONG write-read access.

See also

Text list (Page 210)

ScreenItem Object (Page 134)

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.

0 = solid line

1 = dashed line

2 = dotted line

3 = dash-dotted line

4 = dash-dot-dot line

See also

Text list (Page 210)

ScreenItem Object (Page 134)

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 210)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

1.14.4.11 L

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 271)

ScreenItem Object (Page 134)

LabelColor Property

Description

Defines the color of the scale label.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 210)

ScreenItem Object (Page 134)

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 125)

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:

Code in hexadecimal notation	Description
0x00000000	OK
0x80040001	Execution error
0x80040002	Tag error
0x80040003	Server not available.
0x80040004	Multi Tag Error (Error in one or several tags)

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.

```
'VBS178
```

```
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "LastError: " & group.LastError & vbNewLine
```

The LastError property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "LastError: " & group("Motor1").LastError & vbNewLine
```

See also

TagSet Object (List) (Page 151)

QualityCode Property (Page 487)

ErrorDescription Property (Page 363)

Tag Object (Page 146)

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  " & objScrItem.Layer,vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
```

See also

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

Layer00FillColor property

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 182)

ScreenItem Object (Page 134)

Layer01Value Property

Description

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 182)

ScreenItem Object (Page 134)

Layer02Value Property

Description

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 182)

ScreenItem Object (Page 134)

Layer03Value Property

Description

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 182)

ScreenItem Object (Page 134)

Layer04Value Property

Description

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 134)

3D Bar (Page 182)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 182)

ScreenItem Object (Page 134)

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 140)

Layers Property

Description

Returns an object of type "Layers".

Layers (read-only)

See also

Layers Object (Listing) (Page 129)

HMIRuntime Object (Page 125)

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:

```
'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 575)

ScreenItem Object (Page 134)

LeftComma Property

Description

Defines or returns the number of digits to the left of the decimal point (0 to 20).

See also

Bar (Page 188)

ScreenItem Object (Page 134)

LightEffect Property

Description

TRUE, when the light effect should be activated. BOOLEAN write-read access.

See also

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

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**.

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 265)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

LineWidth Property

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 271)

ScreenItem Object (Page 134)

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**.

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

See also

Text list (Page 210)

ScreenItem Object (Page 134)

Lo

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.

Value	Description
TRUE	Loads archived values are on picture calls.
FALSE	Loads only current values on picture calls.

The attribute can be assigned dynamic properties by means of the name **LoadDataImmediately**.

LoadDataImmediately Property

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

LocaleID Property

Description

Defines the language to be displayed in the control, e.g. 1031 for German. Write/Read access.

The full list of language codes is available in the WinCC documentation (Index > Language ID).

See also

WinCC Slider Control (Page 259)

WinCC Gauge Control (Page 243)

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

LockStatus Property

Description

TRUE, when a locked measuring point should be displayed. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

Logging Property

Description

Returns an object of type "Logging".

Logging (read-only)

See also

HMIRuntime Object (Page 125)

Logging Object (Page 130)

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 188)

ScreenItem Object (Page 134)

LongStrokesOnly Property

Description

TRUE, when only the long sections of a scale should be displayed . BOOLEAN write-read access.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

LongStrokesSize Property

Description

Defines or returns the length of the axis section in pixels.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

LongTermArchiveConsistency property

LongTermArchiveConsistency

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** .

LongTimeArchiveConsistency Property

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 267)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

Slider (Page 223)

3D Bar (Page 182)

ScreenItem Object (Page 134)

MaximizeButton Property

Description

TRUE, when the object can be maximized in Runtime. Read only access.

See also

Picture Window (Page 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

MCGUBackFlash Property

Description

TRUE, when the background should flash when a message departs unacknowledged. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

MCGUTextColorOff 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 208)

ScreenItem Object (Page 134)

MCGUTextColorOn 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 208)

ScreenItem Object (Page 134)

MCGUTextFlash Property

Description

TRUE, when the font should flash when a message departs unacknowledged. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

MCKOBackFlash Property

Description

TRUE, when the background should flash when a message arrives. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

MCKOTextFlash Property

Description

TRUE, when the font should flash when a message arrives. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

MCKQBackFlash Property

Description

TRUE, when the background should flash when a message departs acknowledged. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

MCKQTextFlash Property

Description

TRUE, when the font should flash when a message departs acknowledged. BOOLEAN write-read access.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

MCText Property

Description

Defines or returns the label for the respective message class.

See also

Group Display (Page 208)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 191)

HMIRuntime Object (Page 125)

MessageBlockAlign property

Align message blocks - MessageBlockAlign

Aligns the contents of a selected message block in the table.

The following settings are available:

Value	Description	Description
0	left	Aligns the contents of a selected message block to the left.
1	centered	Aligns the contents of a selected message block to the center.
2	right	Aligns the contents of a selected message block to the right.

The attribute can be assigned dynamic properties by means of the name **MessageBlockAlign**.

MessageBlockAutoPrecisions property

Automatic decimal places - MessageBlockAutoPrecisions

Enables automatic setting of the number of decimal places.

Value	Description
TRUE	The number of decimal places is set automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

The attribute can be assigned dynamic properties by means of the name **MessageBlockAutoPrecisions**.

MessageBlockCaption property

Caption - MessageBlockCaption

Defines the caption of the column header in the message window for the selected message block.

You can only edit the caption after having disabled the "Apply project settings" option.

The caption entered is active in all Runtime languages.

The attribute can be assigned dynamic properties by means of the name **MessageBlockCaption**.

MessageBlockDateFormat property

Date format - MessageBlockDateFormat

Defines the date format for displaying messages.

The following date formats are available:

Value	Description
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.07.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2007.
dd/MM/yy	Day/Month/Year, e.g. 24/12/07.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2007.

The attribute can be assigned dynamic properties by means of the name **MessageBlockDateFormat**.

MessageBlockExponentialFormat property

Exponential notation - MessageBlockExponentialFormat

Specifies the exponential notation for visualization of the values of a selected message block.

Value	Description
TRUE	The values are displayed with exponential notation.
FALSE	The values are displayed with decimal notation.

The attribute can be assigned dynamic properties by means of the name **MessageBlockExponentialFormat**.

MessageBlockFlashOn property

Flashing on - MessageBlockFlashOn

Enables flashing of the selected message block in Runtime after a message was activated.

Value	Description
TRUE	Flashing message block content.
FALSE	No flashing message block content.

The attribute can be assigned dynamic properties by means of the name **MessageBlockFlashOn**.

MessageBlockHideText property

Text content - MessageBlockHideText

Enables the textual display of the content of a selected message block.

Value	Description
TRUE	The content is not displayed as text. The option is disabled.
FALSE	The content is displayed as text. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **MessageBlockHideText**.

MessageBlockHideTitleText property

Text header - MessageBlockHideTitleText

Enables the display of the header of a selected message block in text format.

Value	Description
TRUE	The header is not displayed in text format. The option is disabled.
FALSE	The header is displayed in text format. The option is enabled.

The attribute can be assigned dynamic properties by means of the name **MessageBlockHideTitleText**.

MessageBlockIndex property

MessageBlockIndex

References an existing message block. Using this attribute you can assign the values of other attributes to a specific message block.

Values between 0 and "MessageBlockCount" minus 1 are valid for "MessageBlockIndex". Attribute "MessageBlockCount" defines the number of available message blocks.

The "MessageBlockIndex" attribute can be assigned dynamic properties by means of attribute **MessageBlockRepos**.

Number of Digits - MessageBlockLeadingZeros

Number of digits - MessageBlockLeadingZeros

Defines the number of leading zeros for the message block content. The maximum number is "11". A "0" value disables the "With leading zeros" option.

The attribute can be assigned dynamic properties by means of the name **MessageBlockLeadingZeros**.

MessageBlockLength property

Length in characters - MessageBlockLength

Defines the length of the message block selected based on the number of characters. You can only enter a value if the "Apply project settings" field is disabled.

The attribute can be assigned dynamic properties by means of the name **MessageBlockLength**.

MessageBlockName property

Object name - MessageBlockName

Displays the object name of the message block selected. You cannot edit this name.

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**.

MessageBlockSelected property

Available message blocks - MessageBlockSelected

The available message block can be used in Runtime control 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.

The attribute can be assigned dynamic properties by means of the name **MessageBlockSelected**.

MessageBlockShowDate property

Show date - MessageBlockShowDate

Enables the display of a date in the "Time" message block in addition to the time.

Value	Description
TRUE	Date and time are displayed.
FALSE	The time is displayed.

The attribute can be assigned dynamic properties by means of the name **MessageBlockShowDate**.

MessageBlockShowIcon property

Content as icon - MessageBlockShowIcon

Enables the display of the content of a selected message block as icon.

Value	Description
TRUE	The content is visualized as icon.
FALSE	The content is not visualized as icon.

The attribute can be assigned dynamic properties by means of the name **MessageBlockShowIcon**.

MessageBlockShowTitleIcon property

Header as icon - MessageBlockShowTitleIcon

Enables the display of the header of a selected message block as icon.

Value	Description
TRUE	The header is displayed as icon.
FALSE	The header is not displayed as icon.

The attribute can be assigned dynamic properties by means of the name **MessageBlockShowTitleIcon**.

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.

You can only enter a text ID after having disabled the "Apply project settings" option.

The attribute can be assigned dynamic properties by means of the name **MessageBlockTextId**.

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 208)

ScreenItem Object (Page 134)

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** .

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**.

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** .

MessageColumnRepos property

Up/Down - Message/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**.

MessageColumnSort property

MessageColumnSort

Defines the sorting order of the message block referenced in "MessageColumnIndex" .

The following settings are available:

Value	Description	Description
0	no	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

The attribute can be assigned dynamic properties by means of the name **MessageColumnSort** .

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**.

MessageColumnVisible property

Selected message blocks - Message/HitlistColumnVisible

The list shows the message blocks of the message list or hitlist which are selected for use in Runtime Control.

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**.

MessageListType property

Active list on picture call - MessageListType

Selection field for defining the active list for picture calls.

Value	Description	Description
0	Message list	The currently active messages are displayed after a picture was called.
1	Short-term archive list	A short-term archive list displays the logged messages after the picture was called. The display is updated immediately on activation of new messages.
2	Long-term archive list	A long-term archive list displays the logged messages after a picture was called.
3	Lock list	Only the currently locked messages are displayed after a picture was called.
4	Hitlist	The configured statistics data is displayed after a picture was called.
5	List of messages to be hidden	The messages to be hidden are displayed at the call of a picture.

The attribute can be assigned dynamic properties by means of the name **MessageListType**.

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 223)

Bar (Page 188)

3D Bar (Page 182)

ScreenItem Object (Page 134)

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 pixels / 2) * 0.8 = 40 pixels.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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 pixels * 2 * 0.08 = 6 pixels.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

Moveable Property

Description

TRUE, when the object can be moved in Runtime. Read only access.

See also

Picture Window (Page 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

Moveable Property

Movable

Defines whether the control can be moved in Runtime.

Value	Description
TRUE	The control can be moved in Runtime.
FALSE	The control cannot be moved in Runtime.

The attribute can be assigned dynamic properties by means of the name **Moveable**.

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 265)

ScreenItem Object (Page 134)

MsgFilterSQL Property

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 265)

ScreenItem Object (Page 134)

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**.

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:

```
'VBS160
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
```

Description of DataItem Object

Returns the name of the DataItem object.

See also

ActiveProject Property (Page 278)

WinCC Function Trend Control (before WinCC V7) (Page 267)

Tag Object (Page 146)

Ellipse segment (Page 158)

Layer Object (Page 127)

DataItem Object (Page 119)

NeedleColor Property

Description

Defines or returns the color of the pointer. LONG write-read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

NormalColor Property

Description

Defines the color of the normal area of the scale. LONG write-read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

NumberLines Property

Description

Text list

Defines or return the number of lines the text list object should contain. If the amount of configured text is larger than this value, the selection list receives a vertical scroll bar.

Combobox and list box

Defines or returns for the Combobox and List Box objects the number of entries the object should contain. You can define a maximum of 100,000 lines.

At the same time, the value of the "Number of rows" attribute specifies the upper 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 standard labeling of the new filed 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 210)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

1.14.4.14 O

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 230)

ScreenItem Object (Page 134)

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":

```
'VBS80
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:

```
'VBS81
MsgBox "Screenname: " & HMIRuntime.ActiveScreen.ObjectName
```

See also

Screen Object (Page 140)

ScreenItem Object (Page 134)

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 140)

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 140)

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 140)

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 502)

ScrollPositionX Property (Page 502)

Picture Window (Page 191)

ScreenItem Object (Page 134)

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 502)

ScrollPositionX Property (Page 502)

Picture Window (Page 191)

ScreenItem Object (Page 134)

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

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 134)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

Online property

Refresh started - Online

Enables a refresh of displayed values when calling a picture in Runtime.

Value	Description
TRUE	Enables the refresh of values on picture calls.
FALSE	Disables the refresh of values on picture calls.

The attribute can be assigned dynamic properties by means of the name **Online**.

OnTop Property

Description

TRUE, when the object should remain in the foreground in Runtime. Read only access.

See also

Picture Window (Page 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

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 223)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 223)

Text list (Page 210)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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**.

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**.

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 content 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.

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 content 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.

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 content 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.

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 content 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.

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 content 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**.

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 content 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**.

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 content 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**.

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 content 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.

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 content 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.

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 content 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.

OperatorMessageSourceType1 property

Transfer as - OperatorMessageSourceType1

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType1.

OperatorMessageSourceType2 property

Transfer as - OperatorMessageSourceType2

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType2.

OperatorMessageSourceType3 property

Transfer as - OperatorMessageSourceType3

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType3.

OperatorMessageSourceType4 property

Transfer as - OperatorMessageSourceType4

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType4.

OperatorMessageSourceType5 property

Transfer as - OperatorMessageSourceType5

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType5.

OperatorMessageSourceType6 property

Transfer as - OperatorMessageSourceType6

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name OperatorMessageSourceType6.

OperatorMessageSourceType7 property

Transfer as - OperatorMessageSourceType7

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name **OperatorMessageSourceType7**.

OperatorMessageSourceType8 property

Transfer as - OperatorMessageSourceType8

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name **OperatorMessageSourceType8**.

OperatorMessageSourceType9 property

Transfer as - OperatorMessageSourceType9

Defines the format for transferring the source.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source as text.
1	Value	Transfer the source as value.

The attribute can be assigned dynamic properties by means of the name **OperatorMessageSourceType9**.

OperatorMessageSourceType10 property

Transfer as - OperatorMessageSourceType10

Specifies the format of the source content for the transfer.

The following formats are available:

Value	Description	Description
0	Text	Transfer the source content in text format.
1	Value	Transfer the source content as value.

The attribute can be assigned dynamic properties by means of the name **OperatorMessageSourceType10**.

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 180)

Static text (Page 178)

Text list (Page 210)

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

I/O Field (Page 196)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

OuterBevelWidth Property

Description

Defines the width of the outer bevel in pixels.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 196)

ScreenItem Object (Page 134)

OutputValue Property

Description

Determines 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 210)

I/O Field (Page 196)

ScreenItem Object (Page 134)

1.14.4.15 P

Pa - Pe

PageMode property

Enable paging - PageMode

Enables paging is 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" property to determine the number of messages output per page.

The page up/down buttons of the toolbar can be used if paging is enabled.

Value	Description
TRUE	Paging is enabled for the long-term archive list.
FALSE	Paging is disabled for the long-term archive list.

The attribute can be assigned dynamic properties by means of the name **PageMode**.

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**.

Parent Property

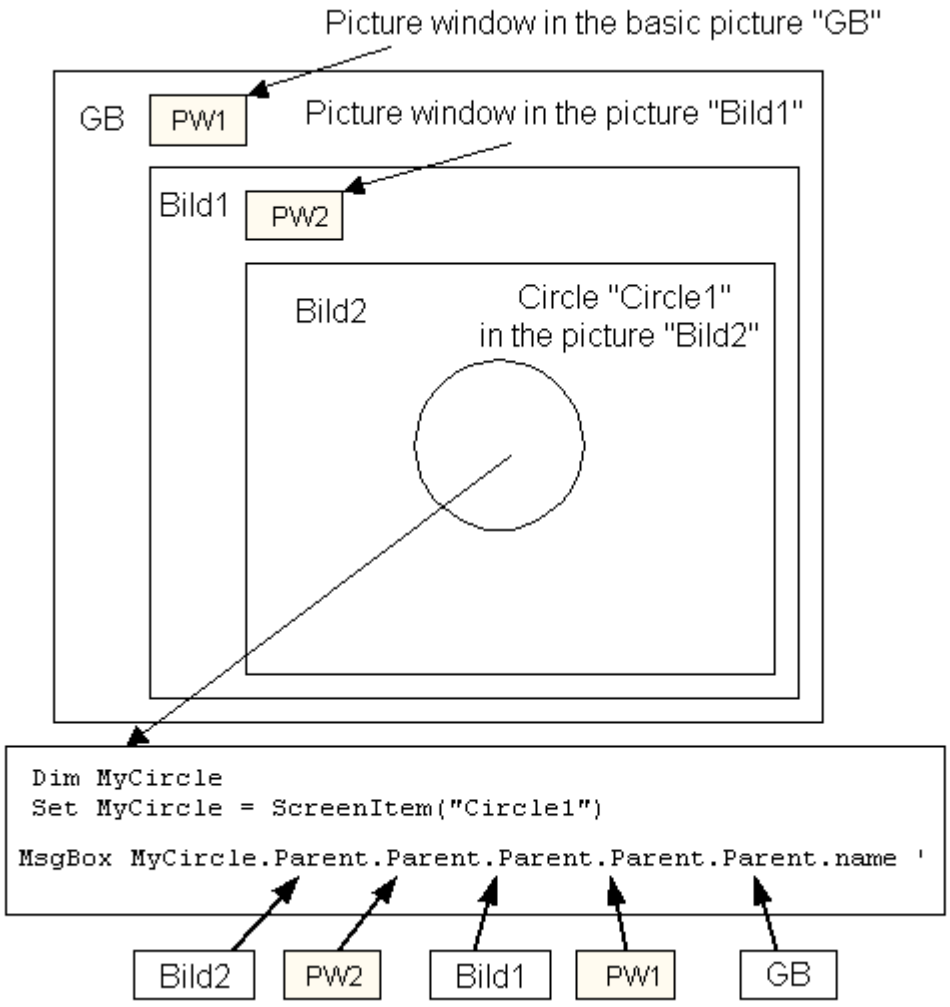
Description

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

```
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:

```
MsgBox MyCircle.Parent.Parent.Parent.Objectname
```

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 example, the object name of the Parent object is displayed:

```
'VBS120
Dim objCircle
Set objCircle = HMIRuntime.Screens("ScreenWindow1").ScreenItems("Circle1")
MsgBox objCircle.Parent.ObjectName
```

See also

Picture Window (Page 191)

Screen Object (Page 140)

Objects and Lists (Page 113)

PasswordLevel Property

Description

Defines the authorization for operation (e.g. no input or no triggering actions) of the object.

See also

ScreenItem Object (Page 134)

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 & vbNewLine
```

See also

Project Object (Page 133)

PercentageAxis property

PercentageAxis

Enables the additional display of an axis with percentage scaling in a trend window for value axes.

Value	Description
TRUE	The display of an axis with percentage scaling is enabled.
FALSE	The display of an axis with percentage scaling is disabled.

The attribute can be assigned dynamic properties by means of the name **PercentageAxis**.

PercentageAxisAlign property

PercentageAxisAlign

Enables axis alignment with percentage scaling in the trend window.

The following settings are available:

Value	Description	Description
0	left	The axis with percentage scaling is aligned left.
1	right	The axis with percentage scaling is aligned right.

The attribute can be assigned dynamic properties by means of the name **PercentageAxisAlign**.

PercentageAxisColor property

PercentageAxisColor

Specifies the color of an axis with percentage scaling. Open the "Color selection" dialog by clicking the button.

The attribute can be assigned dynamic properties by means of the name **PercentageAxisColor**.

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 265)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 265)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

Pi

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 200)

ScreenItem Object (Page 134)

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 200)

ScreenItem Object (Page 134)

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 239)

ScreenItem Object (Page 134)

PictureBack Property

Description

Returns the picture name of the picture for the object background. Read only access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 214)

Round Button (Page 221)

ScreenItem Object (Page 134)

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 221)

Button (Page 214)

ScreenItem Object (Page 134)

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 200)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

PictureThumb Property

Description

Returns the picture name of the background picture for the slider. Read only access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 221)

ScreenItem Object (Page 134)

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 200)

ScreenItem Object (Page 134)

Po - Pr

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 170)

Polygon (Page 168)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 134)

3D Bar (Page 182)

Pressed Property

Description

TRUE, when the RoundButton object is pressed. BOOLEAN write-read access.

See also

Round Button (Page 221)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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**.

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 223)

Radio box (Page 219)

Check box (Page 217)

Bar (Page 188)

3D Bar (Page 182)

ScreenItem Object (Page 134)

ProcessValue property

Description

Returns an object of type "ProcessValue".

See also

Alarms object (list) (Page 116)

ProjectPath Property

Description

Contains the path and name of the associated project.

See also

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 407)

ErrorDescription Property (Page 363)

Tag Object (Page 146)

1.14.4.17 R

Ra - Ri

Radius Property

Description

Defines or returns the radius in pixels.

See also

Pie segment (Page 164)
Circular arc (Page 162)
Circle (Page 160)
Round Button (Page 221)
ScreenItem Object (Page 134)

RadiusHeight Property

Description

Defines or returns the vertical radius in pixels (0 to 5000).

See also

Ellipse segment (Page 158)
Ellipse arc (Page 156)
Ellipse (Page 154)
ScreenItem Object (Page 134)

RadiusWidth Property

Description

Defines or returns the horizontal radius in pixels (0 to 5000).

See also

Ellipse segment (Page 158)

Ellipse arc (Page 156)

Ellipse (Page 154)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

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 259)

ScreenItem Object (Page 134)

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 134)

WinCC Gauge Control (Page 243)

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 166)

Polyline (Page 170)

Polygon (Page 168)

ScreenItem Object (Page 134)

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 134)

Line (Page 166)

Polyline (Page 170)

Polygon (Page 168)

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

RightComma Property

Description

Defines or returns the number of decimal places (0 to 20).

See also

Bar (Page 188)

ScreenItem Object (Page 134)

Ro - Ru

Rotation Property

Description

The icon can be rotated on its middle axis vertical to the character layer. The attribute specifies the angle of rotation in degrees. The angle of rotation is measured counterclockwise.

- 0: The angle of rotation is 0 degrees.
- 90: The angle of rotation is 90 degrees.
- 180: The angle of rotation is 180 degrees.
- 270: The angle of rotation is 270 degrees.

See also

HMI Symbol Library 1.4.1 (Page 233)

ScreenItem Object (Page 134)

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.

0	The standard position of the T-piece is the shape of the letter "T"
90	The "leg" of the "T" points towards the left
180	The "leg" of the "T" points upwards
270	The "leg" of the "T" points to the right

See also

Line (Page 166)

Polyline (Page 170)

Polygon (Page 168)

ScreenItem Object (Page 134)

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 175)

ScreenItem Object (Page 134)

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 134)

RowScrollbar property

Row scroll bars - RowScrollbar

Enables the display of row scroll bars.

The following settings are available:

Value	Description	Description
0	no	No row scroll bars.
1	as required	Row scroll bars are displayed if horizontal space requirements of the control are greater than the actually available display area.
2	always	Row scroll bars are always displayed.

The attribute can be assigned dynamic properties by means of the name **RowScrollbar**.

RowTitleAlign property

Row label alignment - RowTitleAlign

Specifies the type of row label alignment.

The following settings are available:

Value	Description	Description
0	left	The row headers are aligned left.
1	centered	The row headers are aligned to center.
2	right	The row headers are aligned right.

The attribute can be assigned dynamic properties by means of the name **RowTitleAlign**.

RowTitles property

Display row labels - RowTitles

Enables the display of row labels.

Value	Description
TRUE	The row labels are displayed.
FALSE	The row labels are not displayed.

The attribute can be assigned dynamic properties by means of the name **RowTitles**.

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:

Value	Description	Description
0	Discard	The current online configurations are discarded at the next picture change.
1	Retain	The current online configurations are retained at the next picture change.
2	Reset	The current online configurations are reset at the next picture change.

The attribute can be assigned dynamic properties by means of the name **RTPersistence**.

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** .

RTPersistenceType property

Online configuration - RTPersistenceType

Defines how to retain online configurations of WinCC.

The following settings are available:

Value	Description	Description
0	Do not retain	Online configurations are not retained. These are lost at the next picture change.
1	Retain at runtime	Online configurations are retained at runtime. These are lost on exiting.
2	Retain permanently	Online configurations are retained permanently. These are also available after restart.

The attribute can be assigned dynamic properties by means of the name **RTPersistenceType**.

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 271)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

RulerType property

Window - RulerType

Specifies window to be displayed in Runtime. Which specific blocks can be used as columns of the TrendRulerControl depends on the window type.

The following window types can be selected:

Value	Description	Description
0	"Ruler" window	The ruler window shows the coordinate values of the trends on a ruler or values of a selected row in the table.
1	"Statistics range" window	The statistics range window shows the values of the low and high limit of trends between two rulers, or displays the selected range in the table.
2	"Statistics" window	The statistics window shows the statistic evaluation of trends between two rulers, or it displays the selected values in the table.

The attribute can be assigned dynamic properties by means of the name **RulerType**.

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 208)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

Scaling Property

Description

TRUE, when a scale should also be used to represent a value. BOOLEAN write-read access.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 271)

Bar (Page 188)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 191)

ScreenItem Object (Page 134)

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 143)

Screen Object (Page 140)

HMIRuntime Object (Page 125)

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 138)

HMIRuntime Object (Page 125)

ScrollBars Property

Description

TRUE, when the object is equipped with a scroll bar in Runtime. Read only access.

See also

Picture Window (Page 191)

ScreenItem Object (Page 134)

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 134)

OffsetTop Property (Page 458)

OffsetLeft Property (Page 458)

Picture Window (Page 191)

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 458)

OffsetLeft Property (Page 458)

Picture Window (Page 191)

ScreenItem Object (Page 134)

Se

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 134)

WinCC Digital/Analog Clock (Page 239)

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 239)

ScreenItem Object (Page 134)

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 210)

ScreenItem Object (Page 134)

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**.

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**.

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**.

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**.

SelectedTitleColor property

Background selection color - SelectedTitleColor

Specifies the background color of a selected table header. The button opens the "Color selection" dialog.

The settings are only active in Runtime if the "Marking color" option is enabled.

The attribute can be assigned dynamic properties by means of the name **SelectedTitleColor**.

SelectedTitleForeColor property

Font selection color - SelectedTitleForeColor

Specifies the font color of the table header selected. The button opens the "Color selection" dialog.

The settings are only active in Runtime if the "Marking color" option is enabled.

The attribute can be assigned dynamic properties by means of the name **SelectedTitleForeColor**.

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:

Value	Description	Description
0	None	No selection colors for cells and rows.
1	Cell	Selection color for cell.
2	Row	Selection color for row.
3	Cell and row	Selection colors for cell and row.

The attribute can be assigned dynamic properties by means of the name **SelectionColoring**.

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 265)

ScreenItem Object (Page 134)

SelectionRect property

Selection border- SelectionRect

Enables the use of a selection border for selected cells or rows.

The following settings are available:

Value	Description	Description
0	None	No selection border is drawn for selected cells or rows.
1	Cell	A selection border is drawn for the selected cell.
2	Row	A selection border is drawn for the selected row.

The attribute can be assigned dynamic properties by means of the name **SelectionRect**.

SelectionRectColor Property

Description

Specifies the color of the rectangle in the message window if SelectionType equals "1".

See also

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

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**.

SelectionRectWidth Property

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 265)

ScreenItem Object (Page 134)

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**.

SelectionType Property

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 265)

ScreenItem Object (Page 134)

SelectionType property

Selectable lines - SelectionType

Defines the number of lines you can select. The following settings are available:

Value	Description	Description
0	None	No line selection.
1	Single selection	One line can be selected.
2	Multiple selection	Several lines can be selected.

The attribute can be assigned dynamic properties by means of the name **SelectionType**.

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 210)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

ServerNames property

Server selection - ServerNames

Defines which servers within a distributed system should return the display data.

The attribute can be assigned dynamic properties by means of the name **ServerNames**.

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 191)

ScreenItem Object (Page 134)

ServerNames Property

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 265)

ScreenItem Object (Page 134)

Sh - Sk

ShowBar Property

Description

TRUE, when the bar should be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

ShowPosition Property

Description

TRUE, when the slider position is to be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

ShowRuler property

Display ruler - ShowRuler

Enables the display of a ruler for scanning the coordinate points on picture calls.

Value	Description
TRUE	Enables the display of a ruler for scanning the coordinate points.
FALSE	Disables the display of a ruler for scanning the coordinate points.

The attribute can be assigned dynamic properties by means of the name **ShowRuler**.

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 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

ShowRulerInAxis property

ShowRulerInAxis

Enables the display of rulers in the time axis.

Value	Description
TRUE	Enables the display of rulers in the time axes.
FALSE	Disables the display of rulers in the time axes.

The attribute can be assigned dynamic properties by means of the name **ShowRulerInAxis**.

ShowScrollbars property

Scroll bars - ShowScrollbars

Enables the display of scroll bars.

The following settings are available:

Value	Description	Description
0	no	The display of scroll bars is disabled.
1	as required	Scroll bars are displayed if space requirements of the control are greater than the actual display area.
2	always	The scroll bars are always displayed.

The attribute can be assigned dynamic properties by means of the name **ShowScrollbars**.

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.

Value	Description
TRUE	Enables sorting of a selected column by means of sorting button.
FALSE	The sorting button is not displayed.

The attribute can be assigned dynamic properties by means of the name **ShowSortButton** .

ShowSortIcon property

Show sorting icon - ShowSortIcon

Enables the display of the sorting icon.

Value	Description
TRUE	Enables the display of the sorting icon.
FALSE	Disables the display of the sorting icon.

The attribute can be assigned dynamic properties by means of the name **ShowSortIcon**.

ShowSortIndex property

Show Sorting Index - ShowSortIndex

Enables the display of a sorting icon. A sorting index can be displayed if the "0 - up/down/none" entry was selected by clicking it in the "Sorting order on mouse click" field.

Value	Description
TRUE	Enables the display of a sorting index.
FALSE	Disables the display of a sorting index.

The attribute can be assigned dynamic properties by means of the name **ShowSortIndex**.

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.

Value	Description
TRUE	Enables the display of rulers in the statistics field.
FALSE	Disables the display of rulers in the statistics field.

The attribute can be assigned dynamic properties by means of the name **ShowStatisticRuler**.

ShowThumb Property

Description

TRUE, when the slider is to be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

ShowTitle property

Window title - ShowTitle

Defines representation the Control window header.

Value	Designation	Description
0	no	No window title.
1	standard	The window title consists of a WinCC icon and text. The test is entered in the "Text" field.
2	Narrow	The window title consists only of text. The test is entered in the "Text" field.

The attribute can be assigned dynamic properties by means of the name **ShowTitle**.

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**.

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

Sizeable property

Resizable

Enables resizing of the control in Runtime.

For application / picture windows: read-only access.

Value	Description
TRUE	The control can be resized in Runtime.
FALSE	The control cannot be resized in Runtime.

The attribute can be assigned dynamic properties by means of the name **Sizeable**.

SkinName property

Style - SkinName

The control style can be defined in this selection field.

The following settings are available:

Value	Designation	Description
	Project setting	The style corresponds to the project settings in WinCC Explorer.
0	Basic	"Classic" WinCC style
1	Standard	New WinCC V7 style
	Basic Process Control	The style is reserved for internal use with Basic Process Control.

The attribute can be assigned dynamic properties by means of the name **SkinName**.

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 223)

ScreenItem Object (Page 134)

SmartTag property

Description

Returns an object of type "SmartTag".

See also

SmartTags Object (Page 145)

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:

Value	Description	Description
0	Up/down/none	You can toggle between ascending, descending and no sorting by means of mouse click.
1	Up/down	You can toggle between ascending and descending sorting order by means of mouse click.

The attribute can be assigned dynamic properties by means of the name **SortSequence**.

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 267)

ScreenItem Object (Page 134)

SourceControl property

Source - SourceControl

Defines the control to be interconnected with RulerControl.

The attribute can be assigned dynamic properties by means of the name **SourceControl**.

SouceControlType property

Type - SourceControlType

Defines the type of control that is interconnected with the RulerControl in the "Source" field.

The attribute can be assigned dynamic properties by means of the name **SourceControlType**.

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 267)

ScreenItem Object (Page 134)

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 239)

ScreenItem Object (Page 134)

St

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 164)

Circular arc (Page 162)

Ellipse segment (Page 158)

Ellipse arc (Page 156)

ScreenItem Object (Page 134)

State property

Description

Returns the status of a message.

The following table shows the possible states of a message:

State	Alarm Log Status
1	Came In
2	Went Out
5	Came in and comment
6	Gone and comment

See also

Alarms object (list) (Page 116)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

StatusbarBackColor property

Background color - StatusbarBackColor

Defines the background color of the status bar. Open the "Color selection" dialog by clicking the button.

Enable the "Display" option to active the setting.

The attribute can be assigned dynamic properties by means of the name **StatusbarBackColor**.

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** .

StatusbarElementAutoSize property

Automatic - StatusbarElementAutoSize

Enables autosizing of the width of a status bar element selected.

Value	Description
TRUE	The width of the selected element is set automatically.
FALSE	The width of the selected element is not set automatically.

The attribute can be assigned dynamic properties by means of the name **StatusbarElementAutoSize**.

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 measn of the name **StatusbarElementIconId**.

StatusbarElementID property

Object ID - StatusbarElementID

Unique ID of the status bar element selected. WinCC sets this read only ID.

The attribute can be assigned dynamic properties by means of the name **StatusbarElementID**.

StatusbarElementIndex property

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 **StatusbarElementRepos**.

StatusbarElementName property

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 **StatusbarElementRename**.

StatusbarElementRemove property

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 **StatusbarElementRemove**.

StatusbarElementRename property

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 **StatusbarElementRename**. "StatusbarElementRename" also sets a dynamic attribute "StatusbarElementName".

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** .

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**.

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**.

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.

The attribute can be assigned dynamic properties by means of the name **StatusbarElementVisible**.

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**.

StatusbarFontColor property

StatusbarFontColor

Defines the color of the text in the status bar. The attribute can be dynamized with the name **StatusbarFontColor**.

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 265)

ScreenItem Object (Page 134)

StatusbarShowTooltips property

Tooltips - StatusbarShowTooltips

Enables the display of tooltips for the status bar elements in Runtime.

Value	Description
TRUE	Enables the display of tooltips.
FALSE	Disables the display of tooltips.

The attribute can be assigned dynamic properties by means of the name **StatusbarShowTooltips**.

Attribute "StatusbarButtonTooltipText" defines the tooltip text.

StatusbarUseBackColor property

Display background color - StatusbarUseBackColor

Sets a background color for the status bar.

Value	Description
TRUE	Enables the display of the background color of the status bar.
FALSE	Disables the display of a background color for the status bar.

The attribute can be assigned dynamic properties by means of the name **StatusbarUseBackColor**.

StatusbarVisible property

Show status bar - StatusbarVisible

Enables the display of the status bar of a control.

Value	Description
TRUE	Enables the display of a status bar.
FALSE	Disables the display of a status bar.

The attribute can be assigned dynamic properties by means of the name **StatusbarVisible** .

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 1.4.1 (Page 233)

ScreenItem Object (Page 134)

1.14.4.19 T

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**.

TableColor2 property

Row background color 2 - TableColor2

Specifies the background color of "Row color 2". The button opens the "Color selection" dialog.

The settings are only active in Runtime if the "Row color 2" option is enabled. 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**.

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**.

TableForeColor2 property

Row font color 2 - TableForeColor2

Specifies the font color of "Row color 2". The button opens the "Color selection" dialog.

The settings are only active in Runtime if the "Row color 2" option is enabled. 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 **TableForeColor2**.

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 271)

ScreenItem Object (Page 134)

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

The TagPrefix property is not available for the controls.

See also

Picture Window (Page 191)

ScreenItem Object (Page 134)

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 149)

HMIRuntime Object (Page 125)

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 267)

ScreenItem Object (Page 134)

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 134)

Application Window (Page 186)

Text Property

Description

Defines or returns the labeling for an object.

See also

Radio box (Page 219)

Check box (Page 217)

Button (Page 214)

Static text (Page 178)

ScreenItem Object (Page 134)

ThumbBackColor Property

Description

Defines the color of the slider.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

TicColor Property

Description

Defines the color of the scale tick marks. LONG write-read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

TicTextColor Property

Description

Defines the color of the labeling of the scale tick marks.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 134)

WinCC Gauge Control (Page 243)

Ticks Property

Description

TRUE, when the numbered face is displayed. BOOLEAN write-read access.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

TicksColor Property

Description

Defines or returns the color of the hour markings on the face of the analog clock. LONG write-read access.

See also

WinCC Digital/Analog Clock (Page 239)

ScreenItem Object (Page 134)

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").

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

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 271)

ScreenItem Object (Page 134)

TimeAxisActualize property

Refresh - TimeAxisActualize

Enables refreshing of the time axis selected.

Value	Description
TRUE	Enables updates of the trend window which is assigned to the time axis.
FALSE	Disables updates of the trend window which is assigned to the time axis. This setting can be useful when comparing a logged trend with a current trend.

The attribute can be assigned dynamic properties by means of the name **TimeAxisActualize**.

TimeAxisAdd property

New - TimeAxisAdd

Creates a new time axis.

The attribute can be assigned dynamic properties by means of the name **TimeAxisAdd**.

TimeAxisAlign property

Alignment - TimeAxisAlign

Specifies the mode of alignment of a selected time axis.

The following settings are available:

Value	Description	Description
0	Bottom	The time axis selected is displayed below the trend.
1	Top	The time axis selected is displayed above the trend.

The attribute can be assigned dynamic properties by means of the name **TimeAxisAlign**.

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**.

TimeAxisColor property

Time axis color - TimeAxisColor

Specifies the color of the time axis. Use the button to open the "Color selection" dialog.

The setting is only active if the "Use trend color" field is disabled.

The attribute can be assigned dynamic properties by means of the name **TimeAxisColor**.

TimeAxisDateFormat property

Date format - TimeAxisDateFormat

Defines the date format for visualizing a selected time axis.

The following date formats are available:

Value	Description
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.07.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2007.
dd/MM/yy	Day/Month/Year, e.g. 24/12/07.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2007.

The attribute can be assigned dynamic properties by means of the name **TimeAxisDateFormat**.

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**.

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 271)

ScreenItem Object (Page 134)

TimeAxisIndex property

TimeAxisIndex

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**.

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.

Value	Description
TRUE	The trend color is used to display the time axis selected. The setting in the "Color" field is disabled.
FALSE	The time axis selected is displayed using the color set in the "Color" field.

The attribute can be assigned dynamic properties by means of the name **TimeAxisInTrendColor**.

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**.

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**.

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**.

TimeAxisRangeType property

Time range setting - TimeAxisRangeType

Specifies the time range for the time axis selected.

Value	Description	Description
0	Time range	Defines the start time and the time range for the time axis.
1	Start to end time	Defines the start and end time for the time axis.
2	Number of measurement points	Defines the start time and the number of measurement points for the time axis.

The attribute can be assigned dynamic properties by means of the name **TimeAxisRangeType**.

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**.

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".

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. The axis output position is moved away from the trend if the time axis is moved further up in the list and the orientation is the same.

The attribute can be assigned dynamic properties by means of the name **TimeAxisRepos**.

TimeAxisShowDate property

Display date - TimeAxisShowDate

Enables the display of the date and time at the time axis selected.

Value	Description
TRUE	Date and time are displayed. The date format is defined in the "Date format" field.
FALSE	The date is not displayed. Only the time is displayed.

The attribute can be assigned dynamic properties by means of the name **TimeAxisShowDate**.

TimeAxisTimeFormat property

Time format - TimeAxisTimeFormat

Defines the time format for visualizing a selected time axis.

The following time formats are available:

Value	Description
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44.240 PM.

The attribute can be assigned dynamic properties by means of the name **TimeAxisTimeFormat**.

TimeAxisTimeRangeBase property

Time range - TimeAxisTimeRangeBase

Defines the time unit for calculating the time range.

The following time units are available:

Value	Description
500	500 ms
1000	1 second
60000	1 minute
3600000	1 hour
86400000	1 day

The attribute can be assigned dynamic properties by means of the name **TimeAxisTimeRangeBase**.

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**.

TimeAxisTrendWindow property

Trend window - TimeAxisTrendWindow

Specifies the trend window for displaying the time axis selected. Define the available trend windows in the "Trend window" tab.

The attribute can be assigned dynamic properties by means of the name **TimeAxisTrendWindow**.

TImeAxisVisible property

Time axis - TImeAxisVisible

The list shows all time axes you created.

Activate the time axes to be displayed in the trend window in the list.

Click a time axis entry in the list to adapt the properties and to assign the time axis to a trend window.

The attribute can be assigned dynamic properties by means of the name **TimeAxisVisible**.

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 267)

ScreenItem Object (Page 134)

TimeBase property

Time base - TimeBase

This selection field is used to define the time base for the time stamp in the control.

Value	Designation
0	Local time zone
1	Coordinated Universal Time (UTC)
2	Project setting

The attribute can be assigned dynamic properties by means of the name **TimeBase**.

TimeColumn

TimeColumnActualize property

TimeColumnActualize

Enables the update of selected values in the time column.

Value	Description
TRUE	The time column is updated.
FALSE	The time column is not updated. This setting can be useful when comparing tables.

The attribute can be assigned dynamic properties by means of the name **TimeColumnActualize**.

TimeColumnAdd property

New - TimeColumnAdd

Creates a new time column.

The attribute can be assigned dynamic properties by means of the name **TimeColumnAdd**.

TimeColumnAlign property

Alignment - TimeColumnAlign

Defines the mode of alignment of the time column selected.

The following settings are available:

Value	Description	Description
0	left	The time column selected is displayed on the left.
1	Centered	The time column selected is aligned to center.
2	right	The time column selected is displayed on the right.

The attribute can be assigned dynamic properties by means of the name **TimeColumnAlign**.

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 269)

ScreenItem Object (Page 134)

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 disabled
- The "Background color" option is set in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name **TimeColumnBackColor**.

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**.

TimeColumnCaption property

Caption - TimeColumnCaption

Defines the caption of the time column.

The attribute can be assigned dynamic properties by means of the name **TimeColumnCaption**.

TimeColumnDateFormat property

Date format - TimeColumnDateFormat

Defines the date format for visualizing a selected time column.

The following date formats are available:

Value	Description
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.07.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2007.
dd/MM/yy	Day/Month/Year, e.g. 24/12/07.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2007.

The attribute can be assigned dynamic properties by means of the name **TimeColumnDateFormat**.

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**.

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 disabled
- The "Font color" option is set in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name **TimeColumnForeColor**.

TimeColumnHideText property

TimeColumnHideText

Sets text format for displaying the content of a time column.

Value	Description
TRUE	The content is not displayed in text format.
FALSE	The content is displayed in text format.

The attribute can be assigned dynamic properties by means of the name **TimeColumnHideText**.

TimeColumnHideTitleText property

TimeColumnHideTitleText

Sets text format for displaying the time column header.

Value	Description
TRUE	The header is not displayed in text format.
FALSE	The header is displayed in text format.

The attribute can be assigned dynamic properties by means of the name **TimeColumnHideTitleText**.

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**.

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**.

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**.

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**.

TimeColumnRangeType property

Time range setting - TimeColumnRangeType

Defines the time range setting for the time column selected.

Value	Description	Description
0	Time range	Defines the start time and time range of the time column.
1	Start to end time	Defines the start and end time for the time column.
2	Number of measurement points	Defines the start time and the number of measurement points for the time column.

The attribute can be assigned dynamic properties by means of the name **TimeColumnRangeType**.

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**.

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".

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**.

TimeColumnShowDate property

Display date - TimeColumnShowDate

Enables the display of the date and time in the time column selected.

Value	Description
TRUE	Date and time are displayed. The date format is defined in the "Date format" field.
FALSE	The date is not displayed. Only the time is displayed.

The attribute can be assigned dynamic properties by means of the name **TimeColumnShowDate**.

TimeColumnShowIcon property

TimeColumnShowIcon

Enables the display of time column contents as icon.

Value	Description
TRUE	The content is visualized as icon.
FALSE	The content is not visualized as icon.

The attribute can be assigned dynamic properties by means of the name **TimeColumnShowIcon**.

TimeColumnShowTitleIcon property

TimeColumnShowTitleIcon

Enables display of the time column header as icon.

Value	Description
TRUE	The header is displayed as icon.
FALSE	The header is not displayed as icon.

The attribute can be assigned dynamic properties by means of the name **TimeColumnShowTitleIcon**.

TimeColumnSort property

TimeColumnSort

Defines the sorting order of the time column referenced in "TimeColumnIndex" .

The following settings are available:

Value	Description	Description
0	no	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

The attribute can be assigned dynamic properties by means of the name **TimeColumnSort** .

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**.

TimeColumnTimeFormat property

Time format - TimeColumnTimeFormat

Defines the time format for visualizing a selected time column.

The following time formats are available:

Value	Description
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44.240 PM.

The attribute can be assigned dynamic properties by means of the name **TimeColumnTimeFormat**.

TimeColumnTimeRangeBase property

Time range - TimeColumnTimeRangeBase

Defines the time unit for calculating the time range.

The following time units are available:

Value	Description
500	500 ms
1000	1 second
60000	1 minute

Value	Description
3600000	1 hour
86400000	1 day

The attribute can be assigned dynamic properties by means of the name **TimeColumnTimeRangeBase**.

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**.

TimeColumnUseValueColumnColors property

Use value column colors - TimeColumnUseValueColumnColors

Defines whether the selected time column will be displayed in the value column colors.

Value	Description
TRUE	The colors of the value column are used to display a selected time column. The settings in the "Font color" and "Background color" fields are disabled.
FALSE	The colors defined in the "Font color" and "Background color" fields are used to display the selected time column.

The attribute can be assigned dynamic properties by means of the name **TimeColumnUseValueColumnColors**.

TimeColumnVisible property

Time columns - TimeColumnVisible

The list shows the time columns you created. Select the time columns to be displayed in the table from the list.

Click a time column entry in the list to adapt the properties and to define the time range of the time column.

The attribute can be assigned dynamic properties by means of the name **TimeColumnVisible**.

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 269)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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 134)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

ScreenItem Object (Page 134)

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("Tag11")
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 146)

Alarms object (list) (Page 116)

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.

Value	Description
Exact	Only values with precisely the same time stamp are displayed in a table row.
100 ms	All values generated within 100 milliseconds are grouped in a table row.
250 ms	All values generated within 250 milliseconds are grouped in a table row.
500 ms	All values generated within 500 milliseconds are grouped in a table row.
1 s	All values generated within 1 second are grouped in a table row.

The attribute can be assigned dynamic properties by means of the name **TimeStepBase**.

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.

The attribute can be assigned dynamic properties by means of the name **TimeStepFactor**.

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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**.

TitleCut Property

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 265)

ScreenItem Object (Page 134)

TitleCut property

Shorten contents - TitleCut

Truncates the content of column headers if the column is insufficient.

Value	Description
TRUE	The column headers are truncated.
FALSE	The column headers are not truncated.

The attribute can be assigned dynamic properties by means of the name **TitleCut** .

TitleDarkShadowColor property

Dark shading color - TitleDarkShadowColor

Specifies the color of the dark side of shading. The button opens the "Color selection" dialog.

The settings are only active if the "Shading Color" option is enabled.

The attribute can be assigned dynamic properties by means of the name **TitleDarkShadowColor**.

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**.

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**.

TitleLightShadowColor property

Bright shading color - TitleLightShadowColor

Specifies the color of the bright side of shading. The button opens the "Color selection" dialog.

The settings are only active if the "Shading Color" option is enabled.

The attribute can be assigned dynamic properties by means of the name **TitleLightShadowColor**.

Titleline 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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

WinCC Alarm Control (before WinCC V7) (Page 265)

ScreenItem Object (Page 134)

TitleSort property

Sorting by column header - TitleSort

Defines how to trigger sorting by column header. You can only sort by column header if the "Auto Scrolling" option is disabled.

Value	Description	Description
0	no	Sorting by column header is disabled.
1	with click	Sorting is triggered by clicking in the column header.
2	with double-click	Sorting is triggered by double-clicking in the column header.

The attribute can be assigned dynamic properties by means of the name **TitleSort**.

TitleStyle property

Shading color - TitleStyle

Specifies whether to set a shading color for the table header.

Value	Description	Description
0	Flat	Disables the use of shading colors. Flat header style.
1	Button	Enables the use of shading colors. 3D representation of the header.

The attribute can be assigned dynamic properties by means of the name **TitleStyle**.

Toggle Property

Description

TRUE, when the round button should lock after being operated in Runtime. BOOLEAN write-read access.

See also

Round Button (Page 221)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 269)
WinCC Online Trend Control (before WinCC V7) (Page 271)
WinCC Function Trend Control (before WinCC V7) (Page 267)
WinCC Alarm Control (before WinCC V7) (Page 265)
ScreenItem Object (Page 134)

ToolbarAlignment Property

Description

Defines or returns the position of the toolbar. Write/Read access.

See also

WinCC Online Table Control (before WinCC V7) (Page 269)
WinCC Online Trend Control (before WinCC V7) (Page 271)
WinCC Function Trend Control (before WinCC V7) (Page 267)
ScreenItem Object (Page 134)

ToolbarAlignment property

Alignment - ToolbarAlignment

Defines the orientation of the Control toolbar.

The following settings are available:

Value	Description	Description
0	Top	The toolbar is aligned to the top edge.
1	Bottom	The toolbar is aligned to the bottom edge.
2	left	The toolbar is aligned to the left edge.
3	right	The toolbar is aligned to the right edge.

The attribute can be assigned dynamic properties by means of the name **ToolbarAlignment**.

ToolbarBackColor property

Background color - 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 enabled.

The attribute can be assigned dynamic properties by means of the name **ToolbarBackColor**.

ToolbarButtonActive property

Active - ToolbarButtonActive

Activates a button function in Runtime. Clicking the button in Runtime triggers the corresponding function.

Value	Description
TRUE	The button function is enabled.
FALSE	The button function is disabled. You can assign custom functions to the button by means of scripting.

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonActive**.

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** .

ToolbarButtonBeginGroup property

Separator - ToolbarButtonBeginGroup

Enables insertion of a leading separator for a button function selected. These separators can be used to group the icons of the button functions.

Value	Description
TRUE	A separator prefix is inserted for the button function selected.
FALSE	A separator prefix is not inserted for the button function selected.

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonBeginGroup**.

ToolbarButtonClick AlarmControl property

ToolbarButtonClick

Triggers the function linked to the toolbar button. The "ID" can be used by the programmer to call the corresponding button function.

ID	Key function	ID	Key function
1	"Help"	21	"Next message"
2	"Configuration dialog".	22	"Last message"
3	"Message list".	23	"Info text dialog"
4	"Short-term archive list".	24	"Comments dialog"
5	"Long-term archive list"	25	"Loop In Alarm"
6	"Lock List".	26	"Lock message"
7	"Hit List"	27	"Enable message"
8	"List of messages to be hidden"	28	"Hide messages"
9	"Ackn. Central Signaling Devices"	29	'Unhide messages"
10	"Single acknowledgment"	30	"Sort dialog"
11	"Group acknowledgement"	31	"Time base dialog"
18	"Emergency acknowledgement"	32	"Copy lines"
13	"Selection dialog"	33	"Connect backup"
14	"Display options dialog"	34	"Disconnect backup"
15	"Lock dialog"	36	"First page"
17	"Print"	37	"Previous page"
35	"Export data"	38	"Next page"
12	"Autoscroll"	39	"Last page"
19	"First message"	1001	"User-defined 1"
20	"Previous message"		

The attribute can be made dynamic with the name **ToolbarButtonClick**.

ToolbarButtonClick FunctionTrendControl property

ToolbarButtonClick

Triggers the function which is assigned to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

ID	Button function	ID	Button function
1	"Help"	13	"Select time range"
2	"Configuration dialog"	14	"Previous trend"
4	"Zoom area"	15	"Next trend"
5	"Zoom +/-"	16	"Stop"
6	"Zoom X axis +/-"	16	"Start"

7	"Zoom Y axis +/-"	17	"Print"
8	"Shift trend range"	20	"Export data"
9	"Shift axes range"	3	"Ruler"
10	"Original view"	18	"Connect backup"
11	"Select data connection"	19	"Disconnect backup"
12	"Select trends"	1001	"User-defined 1"

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonClick**.

ToolbarButtonClick OnlineTableControl property

ToolbarButtonClick

Triggers the function which is assigned to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

ID	Button function	ID	Button function
1	"Help"	12	"Previous column"
2	"Configuration dialog"	13	"Next column"
3	"First data record"	14	"Stop"
4	"Previous data record"	14	"Start"
5	"Next data record"	15	"Print"
6	"Last data record"	20	"Export data"
7	"Edit"	16	"Define statistics area"
8	"Copy lines"	17	"Calculate statistics"
9	"Select data connection"	18	"Connect backup"
10	"Select columns"	19	"Disconnect backup"
11	"Select time range"	1001	"User-defined 1"

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonClick**.

ToolbarButtonClick OnlineTrendControl property

ToolbarButtonClick

Triggers the function which is assigned to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

ID	Button function	ID	Button function
1	"Help"	17	"Select time range"
2	"Configuration dialog"	18	"Previous trend"
3	"First data record"	19	"Next trend"
4	"Previous data record"	20	"Stop"
5	"Next data record"	20	"Start"

6	"Last data record"	21	"Print"
8	"Zoom area"	26	"Export data"
9	"Zoom +/-"	7	"Ruler"
10	"Zoom time axis +/-"	22	"Define statistics area"
11	"Zoom value axis +/-"	23	"Calculate statistics"
12	"Shift trend range"	24	"Connect backup"
13	"Shift axes range"	25	"Disconnect backup"
14	"Original view"	27	"Relative axis"
15	"Select data connection"	1001	"User-defined 1"
16	"Select trends"		

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonClick**.

ToolbarButtonClick TrendRulerControl property

ToolbarButtonClick

Triggers the function which is assigned to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

ID	Button function
1	"Help"
2	"Configuration dialog"
3	"Ruler window"
4	"Statistics range"
5	"Statistics"
6	"Print"
7	"Export data"
1001	"User-defined 1"

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonClick**.

ToolbarButtonClick UserArchiveControl property

ToolbarButtonClick

Triggers the function which is assigned to the toolbar button. Programmers can use the "ID" to call the corresponding button function.

ID	Button function	ID	Button function
1	"Help"	12	"Read tags"
2	"Configuration dialog"	13	"Write tags"
3	"Select data connection"	14	"Import archive"
4	"First row"	15	"Export archive"

5	"Previous row"	16	"Sort dialog"
6	"Next row"	17	"Selection dialog"
7	"Last row"	18	"Print"
8	"Delete rows"	20	"Export data"
9	"Cut rows"	19	"Time base dialog"
10	"Copy rows"	1001	"User-defined 1"
11	"Insert rows"		

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonClick**.

ToolbarButtonEnabled property

ToolbarButtonEnabled

Enables operation of custom toolbar buttons.

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonEnabled**.

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**.

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**.

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 "ToolbarButtonIndex" 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**.

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**.

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**.

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**.

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**.

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".

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**.

ToolbarButtonTooltipText property

ToolbarButtonTooltipText

Specifies the tooltip text for the button.

The attribute can be assigned dynamic properties by means of the name **ToolbarButtonTooltipText**.

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** .

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 269)
WinCC Online Trend Control (before WinCC V7) (Page 271)
WinCC Function Trend Control (before WinCC V7) (Page 267)
WinCC Alarm Control (before WinCC V7) (Page 265)
ScreenItem Object (Page 134)

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 269)
WinCC Online Trend Control (before WinCC V7) (Page 271)
WinCC Function Trend Control (before WinCC V7) (Page 267)
ScreenItem Object (Page 134)

ToolbarShowTooltips property

Tooltips - ToolbarShowTooltips

Enables the display of tooltips for the button functions in Runtime.

Value	Description
TRUE	Enables the display of tooltips.
FALSE	Disables the display of tooltips.

The attribute can be assigned dynamic properties by means of the name **ToolbarShowTooltips**.

Attribute "ToolbarButtonTooltipText" defines the tooltip text.

ToolbarUseBackColor property

Display background color - ToolbarUseBackColor

Enables the display of the background color for a toolbar.

Value	Description
TRUE	Enables the display of the background color of a toolbar.
FALSE	Disables the display of the background color of a toolbar.

The attribute can be assigned dynamic properties by means of the name **ToolbarUseBackColor**.

ToolbarUseHotKeys property

Hotkeys - ToolbarUseHotKeys

Activates the hotkeys for button functions in Runtime. Insert the hotkeys for button functions in the "Hotkey" field.

Value	Description
TRUE	The hotkeys are activated.
FALSE	The hotkeys are deactivated.

The attribute can be assigned dynamic properties by means of the name **ToolbarUseHotKeys**.

ToolbarVisible property

Display toolbar - ToolbarVisible

Enables the display of the Control toolbar.

Value	Description
TRUE	Enables the display of the toolbar.
FALSE	Disables the display of the toolbar.

The attribute can be assigned dynamic properties by means of the name **ToolbarVisible** .

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 219)
Status display (Page 212)
Connector (Page 180)
Text list (Page 210)
Static text (Page 178)
Slider (Page 223)
Group Display (Page 208)
Rounded rectangle (Page 175)
Round Button (Page 221)
Rectangle (Page 172)
Polyline (Page 170)
Polygon (Page 168)
OLE object (Page 205)
Line (Page 166)
Pie segment (Page 164)
Circular arc (Page 162)
Circle (Page 160)
Group (Page 274)
Graphic Object (Page 200)
Ellipse segment (Page 158)
Ellipse arc (Page 156)
Ellipse (Page 154)
I/O Field (Page 196)
Check box (Page 217)
Button (Page 214)
Bar (Page 188)
Customized object (Page 273)
3D Bar (Page 182)

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 423)

ScreenItem Object (Page 134)

TopConnectedConnectionPointIndex Property

Description

Specifies or sets the index number of the top connecting point.

LONG write-read access.

See also

Connector (Page 180)

ScreenItem Object (Page 134)

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 180)

ScreenItem Object (Page 134)

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 253)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

TrendActualize property

Update -TrendActualize

Enables the update of a selected trend.

Value	Description
TRUE	Enables updates of the trend selected.
FALSE	Disables updates of the trend selected. This setting can be useful when comparing a logged trend with a current trend.

The attribute can be assigned dynamic properties by means of the name **TrendActualize**.

TrendAdd property

New - TrendAdd

Creates a new trend.

The attribute can be assigned dynamic properties by means of the name **TrendAdd**.

TrendAutoRangeBeginTagName property

TrendAutoRangeBeginTagName

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**.

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**.

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**.

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**.

TrendAutoRangeSource property

TrendAutoRangeSource

Defines the mode for automatic calculation of the range of values of trend data.

Value	Description
From display data	The range of values is calculated automatically based on the data displayed.
From project setting	The range of values is defined based on its configured low and high limit. The low and high limits are emulated in the "TrendAutoRangeBeginValue" and "TrendAutoRangeEndValue" attributes.
From online tags.	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 "TrendAutoRangeBeginTagName" and "TrendAutoRangeEndTagName" attributes.

The attribute can be assigned dynamic properties by means of the name **TrendAutoRangeSource**.

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**.

TrendColor Property

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 188)

ScreenItem Object (Page 134)

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**.

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**.

TrendExtendedColorSet property

Extended - TrendExtendedColorSet

Enables configuration of the point and fill colors and the display of colors in Runtime.

Value	Description
TRUE	The "Point color" and "Fill color" field settings can be configured and are active in Runtime.
FALSE	The "Point color" and "Fill color" field settings cannot be configured and are inactive in Runtime.

The attribute can be assigned dynamic properties by means of the name **TrendExtendedColorSet**.

TrendFill property

Filled - TrendFill

Specifies visualization of the area below the trend with filling.

Value	Description
TRUE	The area below the trend is displayed in filled state. You can define the trend color as fill color if the "Advanced" option is deactivated.
FALSE	The trend is not visualized with fill color.

The attribute can be assigned dynamic properties by means of the name **TrendFill**.

TrendFillColor property

Fill color - TrendFillColor

Specifies the fill color of the trend. The settings are only active if the "Filled" option is enabled. Open the "Color selection" dialog by clicking the button.

The settings are only active if the "Advanced" option is enabled.

The attribute can be assigned dynamic properties by means of the name **TrendFillColor**.

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**.

TrendLabel property

Label - TrendLabel

Defines the label of the trend selected. The label is displayed in Runtime if the value at attribute "TrendNameAsLabel" is "FALSE".

The attribute can be assigned dynamic properties by means of the name **TrendLabel**.

TrendLineStyle property

Line style - TrendLineStyle

Defines the line style for trend visualization.

The following settings are available:

Value	Description	Description
0	Solid	The trend is visualized as solid line.
1	Dashed	The trend is visualized as dashed line.
2	Dotted	The trend is visualized as dotted line.
3	Dash dot	The trend is visualized as dot-dash line.
4	Dash Dot Dot	The trend is visualized as dash-dot-dot line.

The attribute can be assigned dynamic properties by means of the name **TrendLineStyle**.

TrendLineType property

Trend type - TrendLineType

Defines how to visualize a trend.

The following settings are available:

Value	Description	Description
0	none	Only the points are displayed.
1	Connect points linearly	Visualizes a trend with linear interconnection of points.
2	Stepped trend	Visualizes a stepped trend and its interconnected points.

The attribute can be assigned dynamic properties by means of the name **TrendLineType**.

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**.

TrendLowerLimit property

What does this attribute define?

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**.

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**.

TrendLowerLimitColoring property

TrendLowerLimitColoring

Enables the "TrendLowerLimitColor" attribute for identifying tag values which are less than the value at "TrendLowerLimitValue".

Value	Description
TRUE	Attribute "TrendLowerLimitColor" is active.
FALSE	Attribute "TrendLowerLimitColor" is inactive.

The attribute can be assigned dynamic properties by means of the name **TrendLowerLimitColoring**.

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**.

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**.

TrendPointColor property

Point color - TrendPointColor

Defines the color of trend points. Open the "Color selection" dialog by clicking the button.

The settings are only active if the "Advanced" option is enabled.

The attribute can be assigned dynamic properties by means of the name **TrendPointColor**.

TrendPointStyle property

Point style - TrendPointStyle

Defines the dot style for trend visualization.

The following settings are available:

Value	Description	Description
0	none	The points are not displayed.
1	Dotted	The trend points are visualized with a size of one pixel. The setting in the "Point width" field is deactivated.
2	Squares	The points are displayed as square. The setting in the "Point width" field is active.
3	Circles	The points are displayed as circles. The setting in the "Point width" field is active.

The attribute can be assigned dynamic properties by means of the name **TrendPointStyle**.

TrendPointWidth property

Point width - TrendPointWidth

Sets the point width in pixels. You can only define the point width for the "square" and "circular" type.

The attribute can be assigned dynamic properties by means of the name **TrendPointWidth**.

TrendProvider property

Data source - TrendProvider

Specifies the data source for a selected trend.

The following settings are available:

Value	Description	Description
0	None	No data source configured for implementation in Runtime by means of script.
1	Archive tags	Data source with archive tags of a process value archive.
2	Online tags	Data source with online tags derived from tag management.
3	User archive	Data source with columns of a user archive.

The attribute can be assigned dynamic properties by means of the name **TrendProvider**.

TrendProviderCLSID_FunctionTrend property

TrendProviderCLSID_FunctionTrend

Indicates the data source of the trend selected.

Value	Description
	No data source configured for implementation in Runtime by means of script.
{416A09D2-8B5A-11D2-8B81-006097A45D48}	Data source with archive tags of a process value archive.
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	Data source with online tags derived from tag management.
{2DC9B1C8-4FC1-41B1-B354-3E469A13FBFD}	Data source with columns of a user archive.

The attribute can be assigned dynamic properties by means of the name **TrendProviderCLSID**.

TrendProviderCLSID_OnlineTrend property

TrendProviderCLSID_OnlineTrend

Indicates the data source of the trend selected.

Value	Description
	No data source configured for implementation in Runtime by means of script.
{416A09D2-8B5A-11D2-8B81-006097A45D48}	Data source with archive tags of a process value archive.
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	Data source with online tags derived from tag management.

The attribute can be assigned dynamic properties by means of the name **TrendProviderCLSID**.

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:

Value	Description	Description
0	Time range	Defines the start time and the time range for the data connection.
1	Start to end time	Defines the start and end time for the data connection.
2	Number of measurement points	Defines the start time and the number of measurement points for the data connection.

The attribute can be assigned dynamic properties by means of the name **TrendRangeType**.

TrendRemove property

Remove - TrendRemove

Removes selected trends from the list.

The attribute can be assigned dynamic properties by means of the name **TrendRemove**.

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".

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**.

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**.

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**.

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**.

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**.

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**.

TrendTimeRangeBase property

Time Range - TrendTimeRangeBase

Defines the time unit for calculating the time range.

The following time units are available:

Value	Description
500	500 ms
1000	1 second
60000	1 minute
3600000	1 hour
86400000	1 day

The attribute can be assigned dynamic properties by means of the name **TrendTimeRangeBase**.

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**.

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**.

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**.

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".

Value	Description
TRUE	The settings of the "TrendUncertainColor" attribute are active.
FALSE	The settings of the "TrendUncertainColor" attribute are inactive.

The attribute can be assigned dynamic properties by means of the name **TrendUncertainColoring**.

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**.

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**.

TrendUpperLimitColoring property

TrendUpperLimitColoring

Enables the "TrendUpperLimitColor" attribute for identifying tag values which are less than the value at "TrendUpperLimit".

Value	Description
TRUE	The setting of the "TrendUpperLimitColor" attribute is active.
FALSE	The setting of the "TrendUpperLimitColor" attribute is inactive.

The attribute can be assigned dynamic properties by means of the name **TrendUpperLimitColoring**.

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**.

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**.

TrendWindow - TrendYAxis

TrendWindowAdd property

New - TrendWindowAdd

Creates a new trend window.

The attribute can be assigned dynamic properties by means of the name **TrendWindowAdd**.

TrendWindowCoarseGrid property

Main scale - TrendWindowCoarseGrid

Enables the display of grid lines for the main scale.

Value	Description
TRUE	Enables the display of grid lines for the main scale.
FALSE	Disables the display of grid lines for the main scale.

The attribute can be assigned dynamic properties by means of the name **TrendWindowCoarseGrid**.

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**.

TrendWindowFineGrid property

Secondary scale - TrendWindowFineGrid

Enables the display of grid lines for the secondary scale.

Value	Description
TRUE	Enables the display of grid lines for the secondary scale.
FALSE	Disables the display of grid lines for the secondary scale.

The attribute can be assigned dynamic properties by means of the name **TrendWindowFineGrid**.

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**.

TrendWindowForegroundTrendGrid property

Only for foreground trend - TrendWindowForegroundTrendGrid

Enables the display of grid lines only for the foreground trend in the trend window.

Value	Description
TRUE	Enables the display of grid lines for the foreground trend in the trend window.
FALSE	Enables the display of grid lines for all trends in the trend window.

The attribute can be assigned dynamic properties by means of the name **TrendWindowForegroundTrendGrid**.

TrendWindowGridInTrendColor property

Use trend color - TrendWindowGridInTrendColor

Sets the trend color for the visualization of the grid lines for the main scale.

Value	Description
TRUE	The grid is displayed in the trend color
FALSE	The grid is displayed with the color set in the "Color" field.

The attribute can be assigned dynamic properties by means of the name **TrendWindowGridInTrendColor**.

TrendWindowHorizontalGrid property

For X axis - TrendWindowVerticalGrid

Enables the display of horizontal grid lines.

Value	Description
TRUE	The display of horizontal grid lines is enabled.
FALSE	The display of horizontal grid lines is disabled.

The attribute can be assigned dynamic properties by means of the name **TrendWindowHorizontalGrid**.

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**.

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**.

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**.

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".

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**.

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 visualization.

The attribute can be assigned dynamic properties by means of the name **TrendWindowRulerColor**.

TrendWindowRulerLayer property

Ruler layer - TrendWindowRulerLayer

Defines the representation layer of a ruler in the trend window.

The following settings are available:

Value	Description	Description
0	Behind grid	The ruler is visualized on a layer behind the grid.
1	Behind trends	The ruler is positioned after the trend and before the grid.
2	In front of trends	The ruler is positioned in front of the trends.

The attribute can be assigned dynamic properties by means of the name **TrendWindowRulerLayer**.

TrendWindowRulerStyle property

Ruler display - TrendWindowRulerStyle

Defines the appearance of the ruler.

The following settings are available:

Value	Description	Description
0	Basic	The ruler is displayed as basic black line.
1	Graphic	The ruler is displayed based on the "color" and "weight" configured.

The attribute can be assigned dynamic properties by means of the name **TrendWindowRulerStyle**.

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 visualization.

The attribute can be assigned dynamic properties by means of the name **TrendWindowRulerWidth**.

TrendWindowSpacePortion property

Space portion - TrendWindowSpacePortion

Specifies the trend window portion for visualization the control.

The attribute can be assigned dynamic properties by means of the name **TrendWindowSpacePortion**.

TrendWindowStatisticRulerColor property

Color of ruler for statistics area - TrendWindowStatisticRulerColor

Specifies the color of the ruler for the statistics area. Open the "Color selection" dialog by clicking the button.

The color can be configured and displayed if "1 - graphic" is set for visualization.

The attribute can be assigned dynamic properties by means of the name **TrendWindowStatisticRulerColor**.

TrendWindowStatisticRulerStyle property

Display ruler for statistics area - TrendWindowStatisticRulerStyle

Enables the display of a ruler for defining the statistics range.

The following settings are available:

Value	Description	Description
0	Basic	The ruler is displayed as basic black line.
1	Graphic	The ruler is displayed based on the "color" and "weight" configured.

The attribute can be assigned dynamic properties by means of the name **TrendWindowStatisticRulerStyle**.

TrendWindowStatisticRulerWidth property

Width of ruler for statistics area - TrendWindowStatisticRulerWidth

Defines the width of the ruler for the statistics area in pixels.

The ruler width can be configured and displayed if "1 - graphic" is set for visualization.

The attribute can be assigned dynamic properties by means of the name **TrendWindowStatisticRulerWidth**.

TrendWindowVerticalGrid property

for Y axis - TrendWindowVerticalGrid

Enables the display of vertical grid lines.

Value	Description
TRUE	The display of vertical grid lines is enabled.
FALSE	The display of vertical grid lines is disabled.

The attribute can be assigned dynamic properties by means of the name **TrendWindowVerticalGrid**.

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**.

TrendXAxis property

X axis - TrendXAxis

Defines the X axis to be used for the trend selected. Define the available X axes inn the "X Axes" tab.

The attribute can be assigned dynamic properties by means of the name **TrendXAxis**.

TrendYAxis property

Y axis - TrendYAxis

Defines the Y axis to be used for the trend selected. Define the available Y axes inn the "Y Axes" tab.

The attribute can be assigned dynamic properties by means of the name **TrendYAxis**.

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":

```
'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 134)

Object Types of the ScreenItem Object (Page 152)

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 134)

Bar (Page 188)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

1.14.4.20 U

Un - Up

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 243)

ScreenItem Object (Page 134)

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 243)

ScreenItem Object (Page 134)

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 is 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 243)

ScreenItem Object (Page 134)

UnitText Property

Description

Defines the text for the unit of measurement. Write/Read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

UnselBGColor Property

Description

Defines or returns the background color of entries in the text list object which are not selected. LONG write-read access.

See also

Text list (Page 210)

ScreenItem Object (Page 134)

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 210)

ScreenItem Object (Page 134)

UpdateCycle Property

Description

Returns the type and frequency of updating the picture window in Runtime. Read only access.

See also

Picture Window (Page 191)

ScreenItem Object (Page 134)

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 271)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

ScreenItem Object (Page 134)

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 134)

WinCC Online Table Control (before WinCC V7) (Page 269)

WinCC Online Trend Control (before WinCC V7) (Page 271)

WinCC Function Trend Control (before WinCC V7) (Page 267)

Us

UseColumnBackColor property

Use column color / background - UseColumnBackColor

Specifies the settings to be activated for the background colors of columns.

Value	Description
TRUE	The background color settings in the "Time columns" and "Value columns" tabs are active.
FALSE	The background color settings are active in the "Display" tab.

The attribute can be assigned dynamic properties by means of the name **UseColumnBackColors**.

UseColumnForeColor property

Use column color / font - UseColumnForeColor

Defines the active font color settings for the columns.

Value	Description
TRUE	The font color settings in the "Time columns" and "Value columns" tabs are active.
FALSE	The font color settings are active in the "Display" tab.

The attribute can be assigned dynamic properties by means of the name **UseColumnForeColors**.

UseMessageColor property

Show message colors - UseMessageColor

Sets the outputs of messages with colors as agreed by handshake.

Value	Description
TRUE	The message colors are displayed.
FALSE	The message colors are not displayed. Instead, the color settings defined for the table content are activated on the "Display" tab.

The attribute can be assigned dynamic properties by means of the name **UseMessageColor**.

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 116)

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 134)

Group Display (Page 208)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

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 208)

ScreenItem Object (Page 134)

UseSelectedTitleColor property

Selection color - UseSelectedTitleColor

Specifies whether to use a selection color for the headers of selected table cells.

Value	Description
TRUE	A selection color is used. The "Background" and "Font" settings are active in Runtime.
FALSE	Selection color is not used. The "Background" and "Font" settings are disabled in Runtime.

The attribute can be assigned dynamic properties by means of the name **UseSelectedTitleColor**.

UseSourceBackColors property

Apply background colors - UseSourceBackColors

Sets the background color derived from the control defined in the "Source" field.

Value	Description
TRUE	The background color from the interconnected control is used.
FALSE	The background color from the interconnected control is not used. The settings on the "Layout" tab are used.

The attribute can be assigned dynamic properties by means of the name **UseSourceBackColors**.

UseSourceForeColors property

Apply font colors - UseSourceForeColors

Sets the font colors derived from the control defined in the "Source" field.

Value	Description
TRUE	The font color of the interconnected control is activated.
FALSE	The font color from the connected control is not used. The settings on the "Layout" tab are used.

The attribute can be assigned dynamic properties by means of the name **UseSourceForeColors**.

UseTableColor2 property

Row Color 2 - UseTableColor2

Specifies whether to use a second row color for the representation of the table.

Value	Description
TRUE	"Row color 2" and "Row color 1" are used alternately.
FALSE	The "Row color 1" settings are used for all rows.

The attribute can be assigned dynamic properties by means of the name **UseTableColor2**.

UseTrendNameAsLabel property

UseTrendNameAsLabel

Sets the "TrendName" or "TrendLabel" attribute for labeling the trend in Runtime.

Value	Description
TRUE	Sets the "TrendName" attribute for labeling the trend in Runtime.
FALSE	Sets the "TrendLabel" attribute for labeling the trend in Runtime.

The attribute can be assigned dynamic properties by means of the name **UseTrendNameAsLabel**.

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 243)

Write Method (Page 725)

Read Method (Page 698)

Tag Object (Page 146)

DataItem Object (Page 119)

ProcessValues Object (List) (Page 132)

ValueAxisAdd property

New - ValueAxisAdd

Creates a new value axis.

The attribute can be assigned dynamic properties by means of the name **ValueAxisAdd**.

ValueAxisAlign property

Alignment - ValueAxisAlign

Specifies the mode of alignment of a selected value axis.

The following settings are available:

Value	Description	Description
0	left	The value axis selected is displayed on left side of the trend.
1	right	The value axis selected is displayed on right side of the trend.

The attribute can be assigned dynamic properties by means of the name **ValueAxisAlign**.

ValueAxisAutoPrecisions property

Decimal places automatic - ValueAxisAutoPrecisions

Enables automatic setting of the decimal precision.

Value	Description
TRUE	The decimal precision is defined automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

The attribute can be assigned dynamic properties by means of the name **ValueAxisAutoPrecisions**.

ValueAxisAutoRange property

Value range automatic - ValueAxisAutoRange

Enables automatic calculation of the range of values.

Value	Description
TRUE	The range of values is calculated automatically.
FALSE	The range of values is calculated based in the values configured in the "from" and "to" fields.

The attribute can be assigned dynamic properties by means of the name **ValueAxisAutoRange**.

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.

The attribute can be assigned dynamic properties by means of the name **ValueAxisBeginValue**.

ValueAxisColor property

Value axis color - ValueAxisColor

Specifies the color of the time axis. Use the button to open the "Color selection" dialog.

The setting is only active if the "Use trend color" field is disabled.

The attribute can be assigned dynamic properties by means of the name **ValueAxisColor**.

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.

The attribute can be assigned dynamic properties by means of the name **ValueAxisEndValue**.

ValueAxisExponentialFormat property

Exponential notation - ValueAxisExponentialFormat

Sets exponential notation for the display of values of a value axis selected.

Value	Description
TRUE	The values are displayed with exponential notation.
FALSE	The values are displayed with decimal notation.

The attribute can be assigned dynamic properties by means of the name **ValueAxisExponentialFormat**.

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**.

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.

Value	Description
TRUE	The selected value axis is displayed in the trend color. The setting in the "Color" field is disabled.
FALSE	The value axis selected is displayed using the color set in the "Color" field.

The attribute can be assigned dynamic properties by means of the name **ValueAxisInTrendColor**.

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**.

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**.

ValueAxisPrecisions property

Decimal places - ValueAxisPrecisions

Specifies the decimal precision for displaying the value axis selected. You can enter the value if the "Automatic" option is disabled.

The attribute can be assigned dynamic properties by means of the name **ValueAxisPrecisions**.

ValueAxisRemove property

Remove - ValueAxisRemove

Removes the selected value axis from the list.

The attribute can be assigned dynamic properties by means of the name **ValueAxisRemove**.

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".

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. The axis output position is moved away from the trend if the value axis is moved further up in the list and the orientation is the same.

The attribute can be assigned dynamic properties by means of the name **ValueAxisRepos** .

ValueAxisScalingType property

Scaling - ValueAxisScalingType

Specifies the scaling mode for a selected value axis.

The following settings are available:

Value	Description	Description
0	Linear	Enables linear scaling of a value axis selected.
1	Logarithmic	Enables logarithmic scaling of a value axis selected.
2	Logarithmically negated	Enables scaling of a selected value value axis with logarithmic negation.

The attribute can be assigned dynamic properties by means of the name **ValueAxisScalingType**.

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**.

ValueAxisVisible property

Value axes - ValueAxisVisible

The list shows all value axes you created.

Activate the value axes to be displayed in the trend windows in the list.

Click a value axis entry in the list to adapt the properties and to assign the value axis to a trend window.

The attribute can be assigned dynamic properties by means of the name **ValueAxisVisible** .

ValueColumn - Vi

ValueColumnAdd property

New - ValueColumnAdd

Creates a new value column.

The attribute can be assigned dynamic properties by means of the name **ValueColumnAdd**.

ValueColumnAlign property

Alignment - ValueColumnAlign

Defines the mode of alignment of a selected value column.

The following settings are available:

Value	Description	Description
0	left	The selected value column is displayed on the left.
1	Centered	The selected value column is aligned to center.
2	right	The selected value column is displayed on the right.

The attribute can be assigned dynamic properties by means of the name **ValueColumnAlign**.

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 269)

ScreenItem Object (Page 134)

ValueColumnAutoPrecisions property

Automatic - ValueColumnAutoPrecisions

Enables automatic setting of the decimal precision.

Value	Description
TRUE	The decimal precision is defined automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

The attribute can be assigned dynamic properties by means of the name **ValueColumnAutoPrecisions**.

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 in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name **ValueColumnBackColor**.

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**.

ValueColumnExponentialFormat property

Exponential notation - ValueColumnExponentialFormat

Sets exponential notation for the display of values of a value column selected.

Value	Description
TRUE	Display with exponential notation.
FALSE	Display with decimal notation.

The attribute can be assigned dynamic properties by means of the name **ValueColumnExponentialFormat**.

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 in the "Use column color" field of the "General" tab.

The attribute can be assigned dynamic properties by means of the name **ValueColumnForeColor**.

ValueColumnHideText property

ValueColumnHideText

Sets text format for displaying the content of a value column.

Value	Description
TRUE	The content is not displayed in text format.
FALSE	The content is displayed in text format.

The attribute can be assigned dynamic properties by means of the name **ValueColumnHideText**.

ValueColumnHideTitleText property

ValueColumnHideTitleText

Sets text format for displaying the value column header.

Value	Description
TRUE	The header is not displayed in text format.
FALSE	The header is displayed in text format.

The attribute can be assigned dynamic properties by means of the name **ValueColumnHideTitleText**.

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**.

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**.

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**.

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" options is disabled.

The attribute can be assigned dynamic properties by means of the name **ValueColumnPrecisions**.

ValueColumnProvider property

Data source - ValueColumnProvider

Specifies the data source for a selected value column.

The following settings are available:

Value	Description	Description
1	Archive tags	Data source with archive tags of a process value archive.
2	Online tags	Data source with online tags derived from tag management.

The attribute can be assigned dynamic properties by means of the name **ValueColumnProvider**.

ValueColumnProviderCLSID property

ValueColumnProviderCLSID

Indicates the data source of the value column selected.

Value	Description
{416A09D2-8B5A-11D2-8B81-006097A45D48}	Data source with archive tags of a process value archive.
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	Data source with online tags derived from tag management.

The attribute can be assigned dynamic properties by means of the name **ValueColumnProviderCLSID**.

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**.

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".

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**.

ValueColumnShowIcon property

ValueColumnShowIcon

Enables the display of value column contents as icon.

Value	Description
TRUE	The content is visualized as icon.
FALSE	The content is not visualized as icon.

The attribute can be assigned dynamic properties by means of the name **ValueColumnShowIcon**.

ValueColumnShowTitleIcon property

ValueColumnShowTitleIcon

Enables display of the value column header as icon.

Value	Description
TRUE	The header is displayed as icon.
FALSE	The header is not displayed as icon.

The attribute can be assigned dynamic properties by means of the name **ValueColumnShowTitleIcon**.

ValueColumnSort property

ValueColumnSort

Defines the sorting order of the value column referenced in "ValueColumnIndex" .

The following settings are available:

Value	Description	Description
0	no	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

The attribute can be assigned dynamic properties by means of the name **ValueColumnSort** .

ValueColumnSortIndex property

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 **ValueColumnSortIndex**.

ValueColumnState property

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 **ValueColumnState**.

ValueColumnTagName property

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 **ValueColumnTagName**.

ValueColumnTimeColumn property

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 **ValueColumnTimeColumn**.

ValueColumnVisible property

Value columns - ValueColumnVisible

The list shows all value columns you created.

Select the value columns to be displayed in the table from the list. Value columns are displayed if interconnected with a time column.

Click a value column entry in the list to adapt the properties, to assign the value column, and to define the data connection.

The attribute can be assigned dynamic properties by means of the name **ValueColumnVisible** .

ValueMax Property

Description

Defines the value at the end of the scale. Write/Read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

ValueMin Property

Description

Defines the value at the start of the scale. Write/Read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

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 269)

ScreenItem Object (Page 134)

VerticalGridLines property

Vertical - VerticalGridLines

Enables the display of vertical dividers.

Value	Description
TRUE	Enables the displays of vertical dividers.
FALSE	Disables the display of vertical dividers.

The attribute can be assigned dynamic properties by means of the name **VerticalGridLines**.

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 134)

Layer Object (Page 127)

HMIRuntime Object (Page 125)

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 243)

ScreenItem Object (Page 134)

WarningColor Property

Description

Defines the color of the "Warning zone" o the scale. LONG write-read access.

See also

WinCC Gauge Control (Page 243)

ScreenItem Object (Page 134)

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.

See also

Bar (Page 188)

ScreenItem Object (Page 134)

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 188)

ScreenItem Object (Page 134)

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":

```
'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 391)

ScreenItem Object (Page 134)

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 191)

Application Window (Page 186)

ScreenItem Object (Page 134)

WindowMonitorNumber property

Description

Defines the number of the monitor on which the picture window is displayed. This requires that your system supports more than one monitor.

The attribute is only effective if the "IndependentWindow" attribute is set to TRUE.

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 223)

Button (Page 214)

ScreenItem Object (Page 134)

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 265)

ScreenItem Object (Page 134)

WithAxes Property

Description

TRUE, when the scale should be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

WithLabels Property

Description

TRUE, when the scale labels should be displayed. BOOLEAN write-read access.

See also

WinCC Slider Control (Page 259)

ScreenItem Object (Page 134)

1.14.4.23 X - Z

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** .

X/YAxisAlign property

Alignment - X/YAxisAlign

Defines the alignment mode for a selected axis.

The following settings are available for the X axis:

Value	Description	Description
0	Bottom	The X axis selected is displayed below the trend.
1	Top	The X axis selected is displayed above the trend.

The X axis attribute can be assigned dynamic properties by means of the name **XAxisAlign**.

The following settings are available for the Y axis:

Value	Description	Description
0	left	The X axis selected is displayed on left side of the trend.
1	right	The X axis selected is displayed on right side of the trend.

The Y axis attribute can be assigned dynamic properties by means of the name **YAxisAlign**.

X/YAxisAutoPrecisions property

Decimal places automatic - X/YAxisAutoPrecisions

Enables automatic setting of the decimal precision.

Value	Description
TRUE	The number of decimal places is set automatically. The value in the "Decimal places" field is disabled.
FALSE	The value in the "Decimal places" field is enabled.

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**.

X/YAxisAutoRange property

Value range automatic - X/YAxisAutoRange

Enables automatic calculation of the value range of the axis selected.

Value	Description
TRUE	The range of values is calculated automatically.
FALSE	The range of values is calculated based in the values configured in the "from" and "to" fields.

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**.

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.

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**.

X/YAxisColor property

Color XY axis - X/YAxisColor

Specifies the color of the axis selected. Open the "Color selection" dialog by clicking the button.

The setting is only active if the "Use trend color" field is disabled.

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**.

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.

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**.

X/YAxisExponentialFormat property

Exponential notation - X/YAxisExponentialFormat

Enables the exponential notation for visualization of a selected axis.

Value	Description
TRUE	The values are displayed with exponential notation.
FALSE	The values are displayed with decimal notation.

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**.

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.

Value	Description
TRUE	The axis selected is displayed in the trend color. The setting in the "Color" field is disabled.
FALSE	The axis selected is displayed using the color set in the "Color" field.

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**.

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**.

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.

X/YAxisPrecisions property

Decimal places - X/YAxisPrecisions

Specifies the decimal precision for displaying the axis selected. You can enter the value if the "Automatic" option is disabled.

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**.

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** .

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** .

X/YAxisScalingType property

Scaling - X/YAxisScalingType

Defines the scaling mode for a selected axis.

The following settings are available:

Value	Description
0	Linear
1	Logarithmic
2	Logarithmically negated

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**.

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**.

X/YAxisVisible property

X/Y axes - X/YAxisVisible

The list shows all axes you created.

Activate the axes to be displayed in the trend windows in the list.

Click an axis entry in the list to adapt the properties and to assign the axis to a trend window.

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** .

XAxisColor Property

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.

XAxisIndex property

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". Attribute "XAxisCount" defines the number of configurable button functions.

The "XAxisIndex" attribute can be assigned dynamic properties by means of attribute **XAxisRepos**.

XAxisRename property

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 **XAxisRename**. "XAxisRename" also sets a dynamic attribute "XAxisName".

YAxisIndex property

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 **YAxisRepos**.

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".

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 134)

Bar (Page 188)

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 188)

3D Bar (Page 182)

ScreenItem Object (Page 134)

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:

```
'VBS97
HMIRuntime.ActiveScreen.Zoom = HMIRuntime.ActiveScreen.Zoom * 2
```

See also

Picture Window (Page 191)

Screen Object (Page 140)

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.

Available Methods in VBS

Activate	GetStatusbarElement	MoveToNextLine	ShowLockDialog
ActivateDynamic	GetStatusbarElementCollection	MoveToNextPage	ShowLockList
Add	GetTimeAxis	MoveToPrevious	ShowLongTermArchiveList
AttachDB	GetTimeAxisCollection	MoveToPreviousLine	ShowMessageList
CalculateStatistic	GetTimeColumn	MoveToPreviousPage	ShowPercentageAxis
CopyRows	GetTimeColumnCollection	NextColumn	ShowPropertyDialog
CreateTagSet	GetToolbarButton	NextTrend	ShowSelectArchive
CutRows	GetToolbarButtonCollection	OneToOneView	ShowSelection
DeactivateDynamic	GetTrend	PasteRows	ShowSelectionDialog
DeleteRows	GetTrendCollection	PreviousColumn	ShowSelectTimeBase
DetachDB	GetTrendWindow	PreviousTrend	ShowShortTermArchiveList
Edit	GetTrendWindowCollection	Print	ShowSort
Export	GetValueAxis	QuitHorn	ShowSortDialog
GetColumn	GetValueAxisCollection	QuitSelected	ShowTagSelection
GetColumnCollection	GetValueColumn	QuitVisible	ShowTimebaseDialog
GetHitlistColumn	GetValueColumnCollection	Read	ShowTimeSelection
GetHitlistColumnCollection	GetXAxis	ReadTags	ShowTrendSelection
GetMessageBlock	GetXAxisCollection	Refresh	StartStopUpdate
GetMessageBlockCollection	GetYAxis	Remove	Stop
GetMessageColumn	GetYAxisCollection	RemoveAll	Trace
GetMessageColumnCollection	HideAlarm	Restore	UnhideAlarm
GetOperatorMessage	Item Method	SelectedStatisticArea	UnlockAlarm
GetOperatorMessageCollection	LockAlarm	ServerExport	Write

GetRulerBlock	LoopInAlarm	ServerImport	WriteTags
GetRulerBlockCollection	MoveAxis	ShowColumnSelection	ZoomArea
GetRulerColumn	MoveToFirst	ShowComment	ZoomInOut
GetRulerColumnCollection	MoveToFirstLine	ShowDisplayOptionsDialog	ZoomInOutTime
GetRulerData	MoveToFirstPage	ShowEmergencyQuitDialog	ZoomInOutValues
GetStatisticAreaColumn	MoveToLast	ShowHelp	ZoomInOutX
GetStatisticAreaColumnCollection	MoveToLastLine	ShowHideList	ZoomInOutY
GetStatisticResultColumn	MoveToLastPage	ShowHitList	ZoomMove
GetStatisticResultColumnCollection	MoveToNext	ShowInfoText	

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
```

See also

ScreenItem Object (Page 134)

Screen Object (Page 140)

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.

Syntax

```
Expression.ActivateDynamic (ByVAl bstrPropertyName As String, ByVal
bstrCycleName As String)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Parameters	Description
bstrPropertyName	Name of property to which trigger relates.
bstrCycleName	Name of activation cycle, e.g. "CycleTime1s".

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

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be added to the list.

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

Parameters	Description
vtName	Name by which value or tag are to be added to list.
vtUserData	Value to be added to list.

Example:

In this example, a value is included in the DataSet list.

```
'VBS172
HMIRuntime.DataSet.Add "Motor1",23
```

See also

TagSet Object (List) (Page 151)

DataSet-Objekt (List) (Page 122)

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

```
VARIANT
```

Parameters	Description
vtApplication	Name of alarm object (optional)

See also

Alarms object (list) (Page 116)

CreateTagSet Method

Function

Creates a new TagSet object. This object may be used for optimized multi-tag access.

syntax

```
Expression.CreateTagSet()
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

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 151)

Tags Object (List) (Page 149)

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

Parameters	Description
bstrPropertyName	Name of property to which trigger relates.

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 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

Parameters	Description
vIndex	Index or name of column of UserArchiveControl.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetColumnCollection method

Function

Returns the collection of all column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
```

```
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetHitlistColumn method

Function

Returns the name or index designated column object of the hitlist of the 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

Parameters	Description
vIndex	Index or name of hitlist column

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetHistlistColumnCollection method

Function

Returns the collection of all column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
```

```
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetMessageBlock method

Function

Returns the name or index designated message block object of the 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

Parameters	Description
vIndex	Index or name of message block.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetMessageBlockCollection method

Function

Returns the collection of all message block objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
```

```
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetMessageColumn method

Function

Returns the name or index designated column object of the AlarmControl as type "ICCAxMessageColumn".

Syntax

```
Expression.GetMessageColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column in message list.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetMessageColumnCollection method

Function

Returns the collection of all column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
```

```
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetOperatorMessage method

Function

Returns the name or index designated operator message block object of the 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

Parameters	Description
vIndex	Index or name of operator message

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetOperatorMessageCollection method

Function

Returns the collection of all operator message block objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
```

```
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetRulerBlock method

Function

Returns the name or index designated block object of the RulerControl as type "ICCAxRulerBlock".

Syntax

```
Expression.GetRulerBlock(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of block in RulerControl

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetRulerBlockCollection method

Function

Returns the collection of all block objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
```

```
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetRulerColumn method

Function

Returns the name or index designated column object of the 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

Parameters	Description
vIndex	Index or name of column of RulerControl.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetRulerColumnCollection method

Function

Returns the collection of all column objects of the RulerControl as type "ICCAxCollection".

Syntax

```
Expression.GetRulerColumnCollection()
```

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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
```

```
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetRulerData method

Function

Returns the value of the called curve at the ruler position of the OnlineTrendControl.

Syntax

```
Ausdruck.GetRulerData(ByVal vTrend As Variant, ByVal RulerIndex As
Long, pvValue As Variant, Optional pvTimeStamp As Variant, Optional
pvFlags As Varian) Long
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

Parameters	Description
vTrend	Name of the curve
RulerIndex	Type of ruler: Ruler, statistic or statistics area
pvValue	Value of X axis
pvTimeStamp	Time
pvFlags	Qualitycode

GetStatisticAreaColumn method

Function

Returns the name or index designated column object of the 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

Parameters	Description
vIndex	Index or name of column of statistics area window.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetStatisticAreaColumnCollection method

Function

Returns the collection of all column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetStatisticResultColumn method

Function

Returns the name or index designated column object of the 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

Parameters	Description
vIndex	Index or name of column of statistics window.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetStatisticResultColumnCollection method

Function

Returns the collection of all column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetStatusbarElement method

Function

Returns the name or index designated operator element of the control status bar as type "ICCAxStatusbarElement".

Syntax

```
Ausdruck.GetStatusbarElement(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of status bar element.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetStatusbarElementCollection method

Function

Returns the collection of all status bar elements of the control as type "ICCAxCollection".

Syntax

```
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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetTimeAxis method

Function

Returns the name or index designated time axis object of the 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

Parameters	Description
vIndex	Index or name of time axis.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetTimeAxisCollection method

Function

Returns the collection of all time axis objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetTimeColumn method

Function

Returns the name or index designated time column object of the 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

Parameters	Description
vIndex	Index or name of time column.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetTimeColumnCollection method

Function

Returns the collection of all time column objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

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

Parameters	Description
vIndex	Index or name of toolbar button function.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetToolbarButtonCollection method

Function

Returns the collection 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 functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetTrend method

Function

Returns the name or index designated curve object of the OnlineTrendControl or FunctionTrendControl as type "ICCAxTrend".

Syntax

```
Ausdruck.GetTrend(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetTrendCollection method

Function

Returns the collection of all curve objects of the OnlineTrendControl or 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetTrendWindow method

Function

Returns the name or index designated curve window object of the OnlineTrendControl or FunctionTrendControl as type "ICCAxTrendWindow".

Syntax

```
Ausdruck.GetTrendWindow(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve window.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetTrendWindowCollection method

Function

Returns the collection of all curve window objects of the OnlineTrendControl or 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetValueAxis method

Function

Returns the name or index designated value axis object of the 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

Parameters	Description
vIndex	Index or name of value axis.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetValueAxisCollection method

Function

Returns the collection of all value axis objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetValueColumn method

Function

Returns the name or index designated value column object of the OnlineTableControl as type "ICCAxValueColumn".

Syntax

```
Ausdruck.GetValueColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of value column of OnlineTableControl.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetValueColumnCollection method

Function

Returns the collection of all value column objects of the OnlineTableControl as type "ICCAxCollection".

Syntax

```
Ausdruck.GetValueColulmnCollection()
```

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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetXAxis method

Function

Returns the name or index designated X axis object of the FunctionTrendControl as type "ICCAxValueAxis".

Syntax

```
Ausdruck.GetXAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of X axis.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetXAxisCollection method

Function

Returns the collection of all X axis objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

GetYAxis method

Function

Returns the name or index designated Y axis object of the FunctionTrendControl as type "ICCAxValueAxis".

Syntax

```
Ausdruck.GetYAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of Y axis.

Example

Use of the "Getxxx" methods in the controls is illustrated on the basis of an example relating to a toolbar button function.

```
'VBS201
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
End Sub
```

Note

If you use the "Get..." methods to access properties of a control, you must use the latter part of the property name. For example, if the index of a toolbar button function is not addressed via ".ToolbarButtonIndex" but via ".Index".

GetYAxisCollection method

Function

Returns the collection of all Y axis objects of the 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

Use of the "GetxxxCollection" methods in the controls is illustrated on the basis of an example relating to the toolbar button functions.

```
'VBS202
Dim ctrl
Set ctrl = ScreenItems.Item("Control1")
Dim coll
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
Dim toolbu
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
Next
End Sub
```

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

- -

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").

NOTICE
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":

```
'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
'
'The objects will be indicate by Item()
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 138)
ScreenItem Object (Page 134)
Tags Object (List) (Page 149)
Alarms object (list) (Page 116)
ProcessValues Object (List) (Page 132)

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

--

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:

Property	Assignment
Value	Tag values
Name	Tag name (unchanged)
QualityCode	Quality level
Timestamp	Current tag time stamp
LastError	0
ErrorDescription	" "

If the value of the tag is not read successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	VT_Empty
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	Read operation error codes
ErrorDescription	Error description on LastError

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:

Parameters	Description
0	The tag value is read from the process image (cache). 0 is the default value.
1	The value of a tag is read directly from AS or channel (direct).

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.

```
'VBS174
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 741)

Example: Writing tag values (Page 739)

LastError Property (Page 407)

ErrorDescription Property (Page 363)

TagSet Object (List) (Page 151)

Tag Object (Page 146)

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 140)

Screens Object (List) (Page 143)

HMIRuntime Object (Page 125)

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

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be removed from the list.

Example:

The following example shows how several tags are included in the TagSet list, and how to remove a tag again.

```
'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

VARIANT

Parameters	Description
Name	Name of the object to be removed from the list.

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:

Assigned Value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

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
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("2004-08-22","2004-09-22",-1) &
vbNewLine
```

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 732)

Example: How to Start an Action on the Server (Logging Object) (Page 746)

Logging Object (Page 130)

DataSet-Objekt (List) (Page 122)

DataLogs Object (Page 121)

AlarmLogs Object (Page 118)

TagSet Object (List) (Page 151)

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-Objekt (List) (Page 122)

TagSet Object (List) (Page 151)

Tag Object (Page 146)

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:

Assigned Value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

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) &
vbNewLine
```

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) & vbNewLine
```

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
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Restore("","2004-09-20",-1) &
vbNewLine
```

See also

Error Messages from Database Area (Page 732)

Example: How to Start an Action on the Server (Logging Object) (Page 746)

Logging Object (Page 130)

DataLogs Object (Page 121)

AlarmLogs Object (Page 118)

1.14.5.6 Methods S to T

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

- -

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:

```
'VBS124
HMIRuntime.Stop
```

See also

HMIRuntime Object (Page 125)

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:

```
'VBS103
HMIRuntime.Trace "Customized error message"
```

See also

HMIRuntime Object (Page 125)

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

- -

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:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	0
ErrorDescription	" "

If the value of the tag is not set successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	Write operation error codes
ErrorDescription	Error description on LastError

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.

Parameters	Description
Value (optional)	The tag value is specified. The specified value overwrites the value in the "Value" property in the tag object. The tag value is not specified. The tag receives the current value from the "Value" property of the tag object.
Writemode (optional)	0 or empty: The tag value is written asynchronously. 0 is the default value. 1: The tag value is written synchronously.

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
```

If you set the optional parameter "Writemode" equal to 1, the process tags are written synchronously (directly to AS).

```
group.Write 1
```

See also

LastError Property (Page 407)

ErrorDescription Property (Page 363)

TagSet Object (List) (Page 151)

Tag Object (Page 146)

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

```
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

```
Ausdruck.ZoomMove()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

- -

1.14.6 Appendix

1.14.6.1 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:

0x0	OK
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):

Error code	Error Message
0x80047200	WinCC is not activated
0x80047201	Invalid archive type
0x80047202	Invalid lower boundary
0x80047203	Invalid upper boundary
0x80047204	Path 'CommonArchiving' could not be created in the project path
0x80047205	Timeout, please retry
0x80047206	WinCC was deactivated
0x80047207	Wrong signification At least one database had a 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 703)
Write Method (Page 725)
Read Method (Page 698)
Restore Method (Page 708)
Logging Object (Page 130)
DataLogs Object (Page 121)
AlarmLogs Object (Page 118)

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 734)

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

See also

Example: Starting an external application (Page 760)

Example: Writing Object Properties (Page 743)

Example: How to Read Tag Values (Page 741)

Example: Writing tag values (Page 739)

Example: Configuring diagnostics output via Trace (Page 738)

Example: Configuring Change Picture Via Property (Page 738)

Example: Configuring change picture globally (Page 737)

Example: Deactivating Runtime (Page 737)

Example: How to Configure Language Changes (Page 736)

Example: Defining the color of objects (Page 736)

Example: Accessing objects in Graphics Designer (Page 735)

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 734)

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 734)

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 734)

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 734)

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 734)

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 734)

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"
```

See also

Examples in WinCC (Page 734)

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:

```
'VBS131
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 8.1
```

or

```
'VBS132
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 8
objTag.Write ,1
```

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
```

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 725)

Examples in WinCC (Page 734)

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.

See also

Read Method (Page 698)

Examples in WinCC (Page 734)

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:

```
'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:

```
'VBS146
Function BackColor_Trigger(ByVal Item)
BackColor_Trigger = RGB(125,0,0)
End Function
```

NOTICE
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 109)

Examples in WinCC (Page 734)

1.15.2.12 Example: How to Start an Action on the Server (Logging Object)

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
HMIRuntime.Tags("StartLogging").Write 0,1
Elseif (StartLogging = 2) Then
RetVal = HMIRuntime.Logging.Remove(TimeFrom, TimeTo, -1)
HMIRuntime.Tags("RetVal").Write RetVal, 1
HMIRuntime.Tags("StartLogging").Write 0,1
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 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 FuntionTrendControl "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".

```
'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".

```
'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 trend control function named "Control1" with the values which describe a parabola.

```
'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 Function Trend Control 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".

```
'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.

```
'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.

See also

General examples for VBScript (Page 755)

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

1. Create a new process picture in the Graphics Designer.
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 "OnToolbarButtonClicked" object event, configure the VB script "Create VBS action on "OnToolbarButtonClicked" event of user-defined toolbar button (VBS302)".
13. Close the "Object properties" dialog box.

Creating a process picture for the selection dialog

1. Create a new process picture in the Graphics Designer.
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.
4. Under "Geometry", set value "200" for the "Picture width" and "Picture height" attributes.
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 "OnToolbarButtonClicked" 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 PM")
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

1. Create a new process picture in the Graphics Designer.
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.

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 756)
Example: Starting an external application (Page 760)
Example: Using the MS Automation interface (Page 758)
Example: Calling Methods of an ActiveX Control (Page 748)

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.
3. Programming.

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.
4. Programming.

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")
objConnection.ConnectionString = strConnectionString
objConnection.Open
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
objRecordset.movefirst
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 755)

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.DoCmd.OpenReport "RPT_WINCC_DATA", 2
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 755)

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 755)

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

ANSI-C for Creating Functions and Actions 2

2 Resources

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

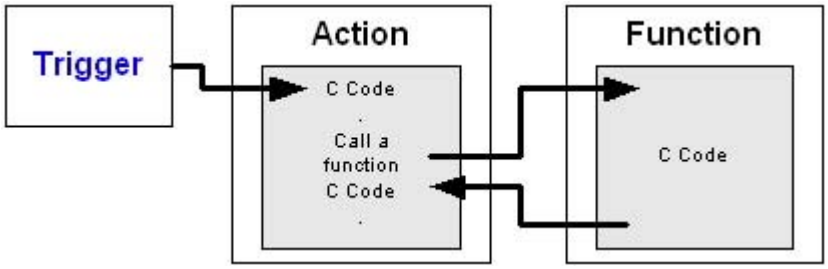
2.2 Creating Functions and Actions

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.

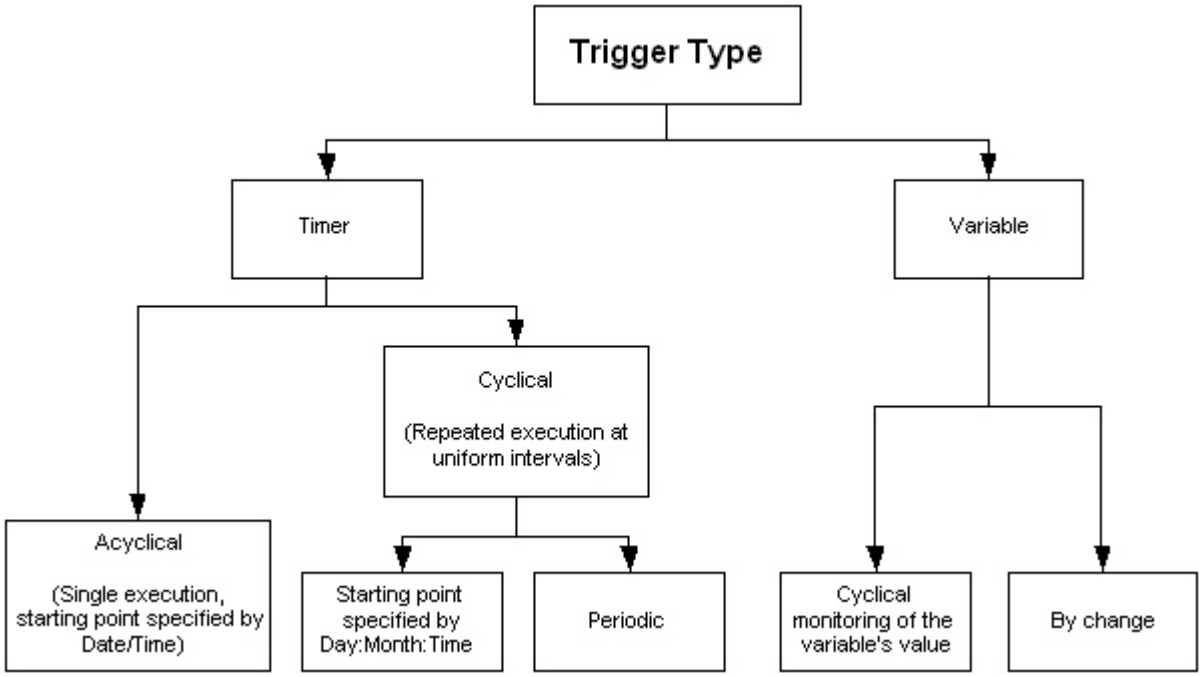
Difference between Functions and Actions

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.



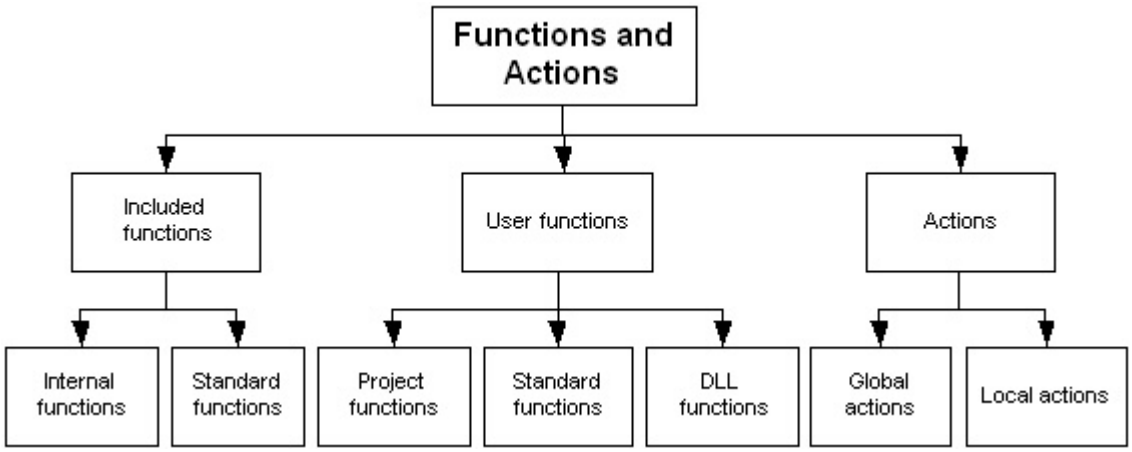
Trigger Types

The following trigger types are available:



Outline of the Functions and Actions

The diagram provides an overview of the range of functions and actions:



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 supports the design, creation and editing of functions and actions with the Global Script editor. Global Script is started from the navigation window of WinCC Explorer.

See also

Runtime Behavior of Actions (Page 824)
How To Create and Edit Actions (Page 802)
Creating and Editing Functions (Page 790)
The Global Script Editor (Page 775)
Use of DLLs in Functions and Actions (Page 774)
Use of Global C-Tags (Page 772)
How to Add Global Script Runtime to a Project's Startup List (Page 771)
How to Generate a New Header (Page 786)
Characteristics of Global Actions (Page 770)
Characteristics of Local Actions (Page 769)
Characteristics of Internal Functions (Page 768)
Characteristics of Standard Functions (Page 767)
Characteristics of Project Functions (Page 766)

2.3 Characteristics of Project Functions

Characteristics of Project Functions

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

Project functions can be used...

- 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 798)
Creating and Editing Functions (Page 790)

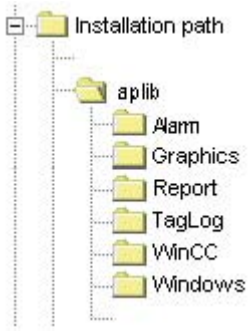
2.4 Characteristics of Standard Functions

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

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 796)

Creating and Editing Functions (Page 790)

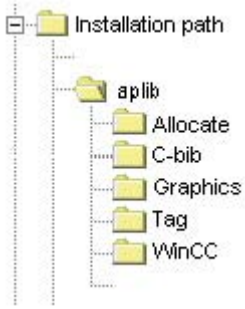
2.5 Characteristics of Internal Functions

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 "\aplib" 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

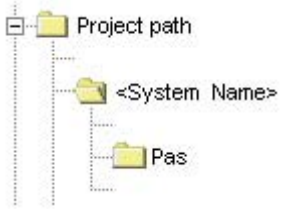
2.6 Characteristics of Local Actions

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 809)

Triggers (Page 811)

How To Create and Edit Actions (Page 802)

How to Add Global Script Runtime to a Project's Startup List (Page 771)

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 809)

Triggers (Page 811)

How To Create and Edit Actions (Page 802)

How to Add Global Script Runtime to a Project's Startup List (Page 771)

2.8 How to Add Global Script Runtime to a Project's Startup List

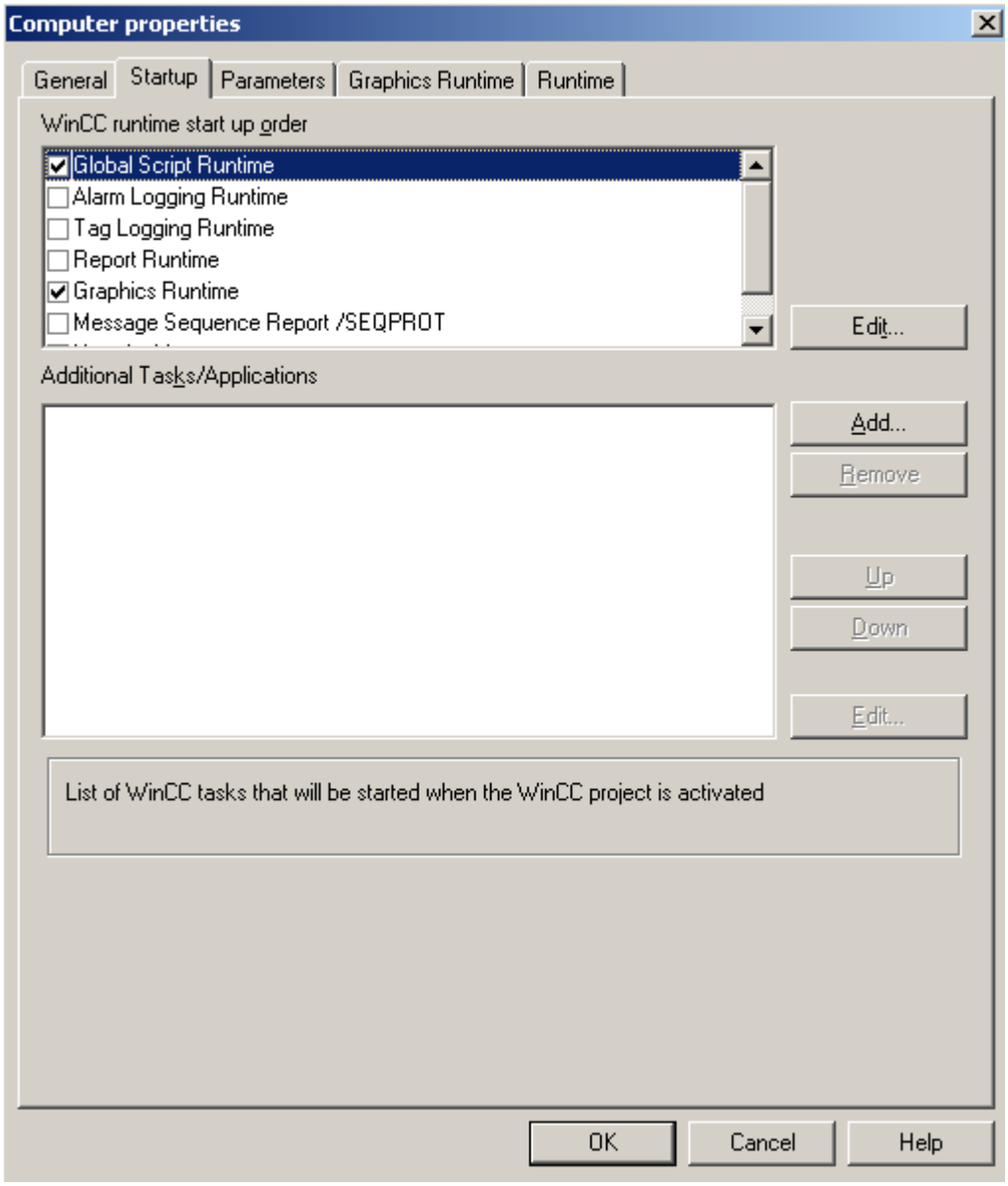
Introduction

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 Properties" dialog is opened.
2. Click the "Startup" tab.

3. Selection option "Global Script Runtime".



4. 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:

```
int a; //The tag a is defined as an integer
```

```
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:

```
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:

```
#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:

```
#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.

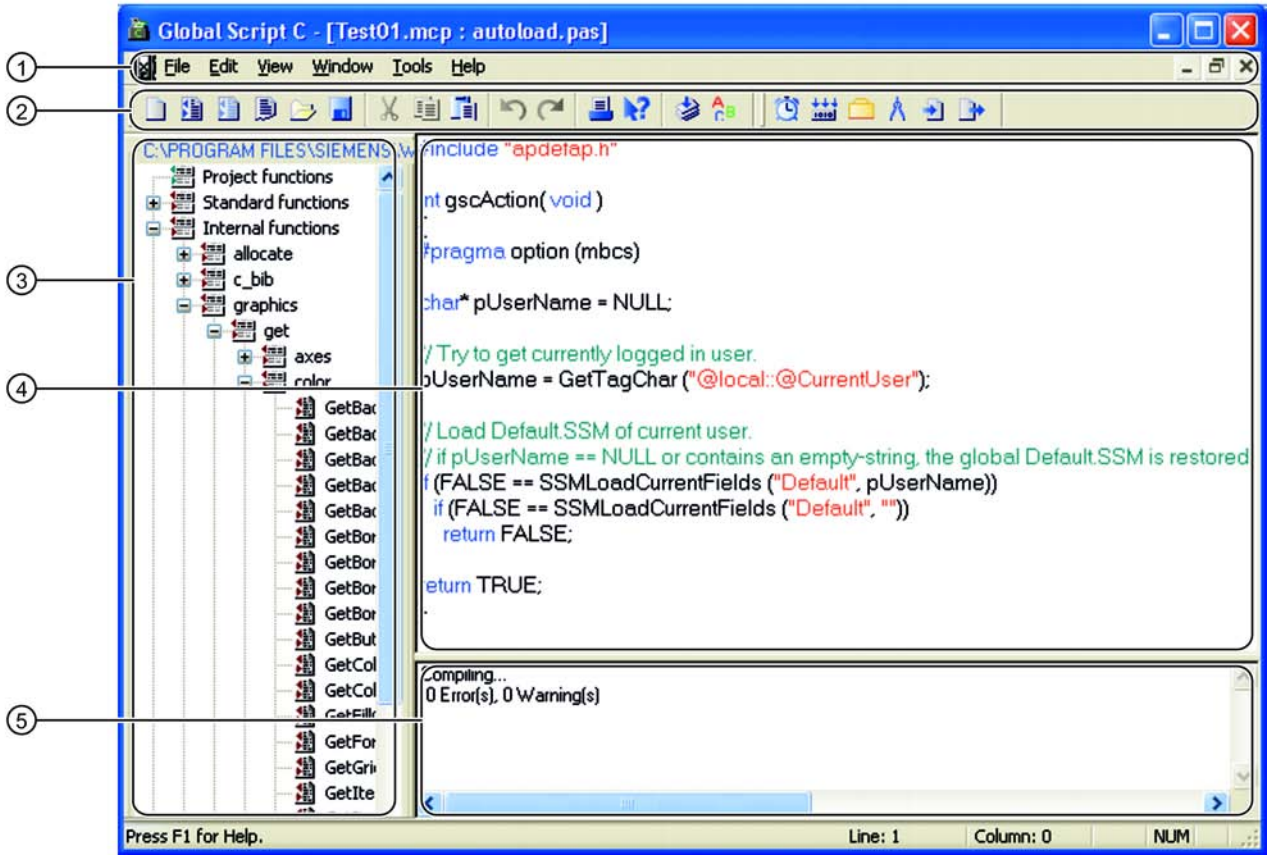


Figure 2-1 The Global Script editor

The Navigation Window

The navigation window is used to select functions and actions to be edited or to be inserted at the location of the cursor in an edit window

The functions and actions are organized in a hierarchical system of groups.

Functions are displayed with their function name. For actions, the file name is displayed.

The Edit Window

Functions and actions are written and edited in an edit window. 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 edit windows can be open at the same time.

The Output Window

The results of the functions "Find in Files" or "Compile All Functions" are displayed in the output window. By default, it is visible, but can be hidden.

Find in Files:

The results of the search are displayed in the output window with one line per search term found. Each line has a line number and shows the path and file name plus the line number and text of the line in which the search term was found.

By double-clicking a line displayed in the output window, you can open the associated file. The cursor is placed in the line in which the search term was found.

Compile All Functions:

If necessary, the compiler outputs warnings and error messages for each compiled function. In the next line, the path and file name of the compiled function as well as the summary message from the compiler are displayed.

The Menu Bar

The menu bar buttons vary, depending on the situation. It is always visible.

The Toolbars

Global Script has two toolbars. They can be made visible as needed and dragged to any position on the screen with the mouse.

The Status Bar

The status bar is located at the lower edge of the Global Script window and can be shown or hidden. 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.

See also

Printing Functions and Actions (Page 788)
How to Search in Files (Page 787)
How to Compile All Functions (Page 786)
How to Generate a New Header (Page 786)
How to Delete Actions or Project and Standard Functions (Page 785)
How to Use "Save As..." (Page 784)
How to Set the Font Style (Page 784)
How to Set Different Views (Page 783)
Working with the Toolbars (Page 781)
Working in the Edit Window (Page 778)

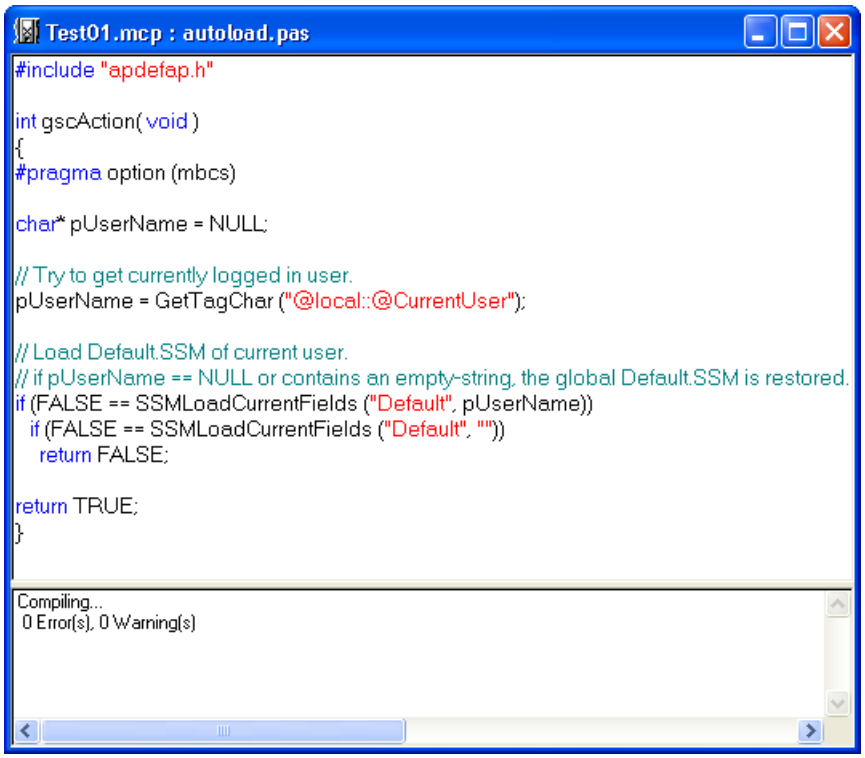
2.11.2 Working in the Edit Window

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.



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:

Color	Description	Example
Blue	Keywords	define, double, if
Green	Comments	// is a comment
Red	Strings	"Rectangle3"
Black	Other C-codes	level=100*newvalue/255;

Note

A function or action cannot have more than 32767 characters including spaces.

See also

Editing Functions with the Mouse (Page 780)

Editing Functions with the Keyboard (Page 780)

2.11.2.2 Editing Functions with the Keyboard

You can carry out the following editing functions using the keyboard:

Editing function	Keyboard operation
Switch between write modes Insert/Overwrite	<INSERT>
Add new line	<ENTER>
Delete one character to the right	<DELETE>
Delete one character to the left	<BACKSPACE>
Delete marked text	<DELETE> or <BACKSPACE>
Jump to beginning of line	<POS1>
Jump to end of line	<END>
Jump to beginning of text	<CTRL+POS1>
Jump to end of text	<CTRL+END>
Move cursor	<Cursor keys>
Move cursor by one window content to beginning of text	<PAGE UP>
Move cursor by one window content to end of text	<PAGE DOWN>
Move cursor to first line in window	<CTRL+PAGE UP>
Move cursor to last line in window	<CTRL+PAGE DOWN>
Jump to next tab position	<TAB>
Cut marked text and paste to clipboard	<CTRL+X>
Copy marked text to clipboard	<CTRL+C>
Paste text from clipboard	<CTRL+V>

2.11.2.3 Editing Functions with the Mouse

You can carry out the following editing functions using the mouse:

Editing function	Mouse command (left mouse button)
Select text	Drag mouse over text
Select a word	Double-click the word
Select a line	Triple-click the line

Editing function	Mouse command (left mouse button)
Extended selection	<SHIFT> + mouse-click
Set cursor	Click
Move selected text	Drag and drop
Duplicate selected text	<CTRL>+drag and drop

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

2.11.3 Working with the Toolbars

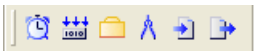
Purpose

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:

The standard toolbar

The editing toolbar



Contents

The standard toolbar contains buttons for the following functions:

Button	Function	Key combination
	Creates a new action.	<ALT+A> or <CTRL+N>
	Creates a new standard function.	<ALT+S> or <CTRL+N>

Button	Function	Key combination
	Creates a new project function.	<ALT+P> or <CTRL+N>
	Creates a new header file.	<CTRL+H>
	Opens an existing action or function.	<CTRL+O>
	Saves the content of the active edit window. This function is only available, if an edit window is open.	<CTRL+S>
	Cuts the selected text and copies it to the clipboard. This function is only available, if text has been selected.	<CTRL+X>
	Copies the selected text to the clipboard. This function is only available, if text has been selected.	<CTRL+C>
	Pastes the contents of the clipboard at the location of the cursor. This function is only available, if the clipboard is not empty.	<CTRL+V>
	Undoes the last maximum 30 editor actions. This function is only available, if an editor action has been performed.	<CTRL+Z>
	Redoes the last editor action that was undone. This function is only available, if an editor action has been undone.	<CTRL+A>
	Prints the contents of the active edit window as project documentation. This function is only available, if an edit window is open.	<CTRL+P>
	Activates the direct help (What's this?).	<SHIFT+F1>
	Generates a new header.	<CTRL+G>
	Allows you to set the font.	<CTRL+F>

Note

The key combination <CTRL+N> is only available, if at least one edit window is open. If the active edit window contains a function and <CTRL+N> is pressed, a new project function is created. If the active edit window contains an action, this key combination creates a new global action.

The editing toolbar contains buttons for the following functions:

Button	Function	Key combination
	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 edit window is open.	<CTRL+I>
	Compiles the code in the active edit window. This function is only available, if an edit window is open.	<SHIFT+F8>
	Opens the tag dialog. This function is only available, if an edit window is open.	<CTRL+R>

Button	Function	Key combination
	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 edit window. This function is only available, if an edit window is open.	<CTRL+W>
	Imports an action. This function is only available, if there is an action in the active window.	<CTRL+M>
	Exports the action from the active edit window. This function is only available, if there is an action in the active window.	<CTRL+T>

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 783)

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 785)

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 subdirectory).

When the header is generated again, the functions that have been copied are entered into the corresponding header files and are then available to you to use in your project.

Requirement

This function is only available, if all edit windows are closed.

Procedure

1. Click the button 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

After all manual changes have been made to the header files, all functions must be recompiled. All project, standard and internal functions are compiled automatically.

The results of the individual compilations are displayed in the output window as follows:

If necessary, the compiler outputs warnings and error messages for each compiled function. In the next line, the path and file name of the compiled function as well as the summary message from the compiler are displayed.

If functions are called in other functions, errors might occur and error messages might be displayed. 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>.

Note

In a multi-user project, the "Compile All Functions" function is not available. The reason for this is that it is no longer possible to allocate the functions in such projects.

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 789)

How to Open Page View (Page 789)

How to Set the Print Parameters (Page 788)

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.

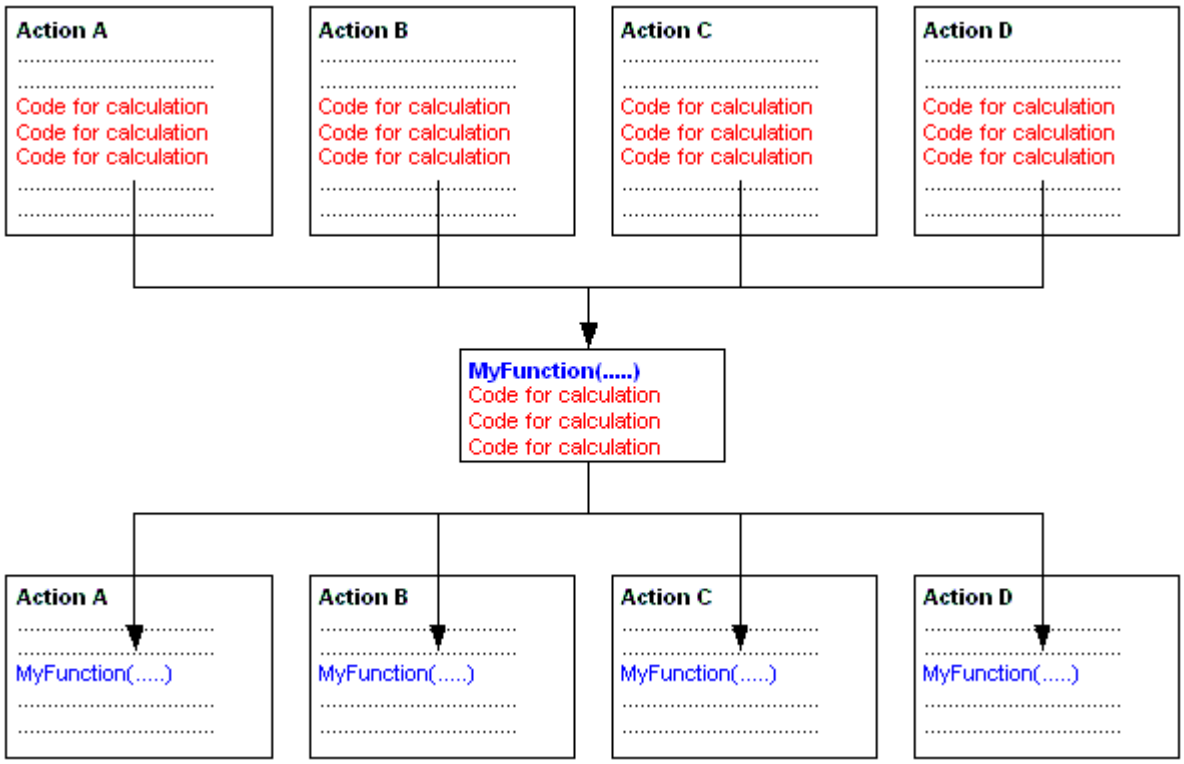
Internal functions cannot be created or edited.

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:

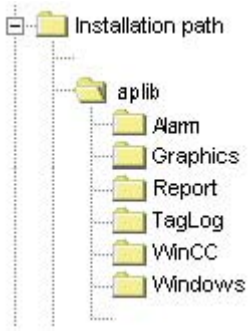
- The code is only programmed once.
- Modifications are only made at one point, namely in the procedure, not in each action.
- The action code is shorter and, thus clearer.



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:



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 778)

How to Use Functions from Other Sources (Page 801)

How to Rename a Function (Page 800)

How to Compile and Save a Function (Page 799)

How to Protect a Function Against Unauthorized Access (Page 798)

Inserting Additional Function-Related Information (Page 797)

How to Use Standard and Project Functions (Page 796)

How to Use Internal Functions (Page 795)

How to Write Function Code (Page 794)

How to Create a New Function (Page 793)

Characteristics of Standard Functions (Page 767)

Characteristics of Project Functions (Page 766)

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

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. In the navigation window, open the shortcut menu for the desired group.
2. Select "New".
   If a new function is created, the first line of code in the associated edit window contains the type of value returned and the default name of the new function. In the following brackets, you can enter a formal parameter if you wish.
   The function code is entered between the braces.

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 797)

How to Use "Save As..." (Page 784)

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 796)

How to Use Internal Functions (Page 795)

Editing Functions with the Mouse (Page 780)

Editing Functions with the Keyboard (Page 780)

Working in the Edit Window (Page 778)

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.

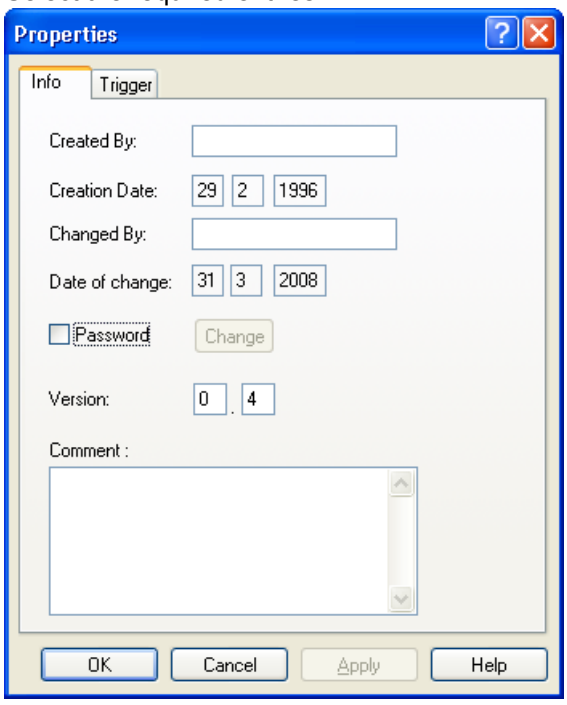
When a new action is created, the creation date is automatically entered in the function-related information and is unchangeable. In addition, the version number 1.0 is also assigned to the number. The version numbers can be individually assigned when editing a function. When a function is changed and 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.

Requirement

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 798)

How to Set Different Views (Page 783)

Working with the Toolbars (Page 781)

2.12.7 How to Protect a Function Against Unauthorized Access

Introduction

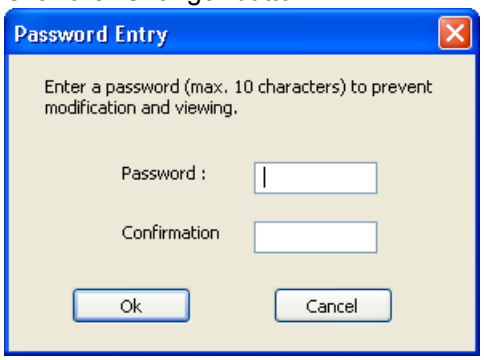
Functions can be protected with a password against unauthorized read and write access. The password is a part of the function-related information.

Requirement

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 the entries by clicking "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 783)

Working with the Toolbars (Page 781)

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 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.

Requirement

The function to be compiled must be opened in the edit window.

Procedure

1. Click the button in the editing toolbar.
2. Examine the compiler messages in the lower portion of the edit window.
3. 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.

4. 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.
5. Click the button in the standard 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.

See also

Runtime Behavior of Actions (Page 824)

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 the button 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.

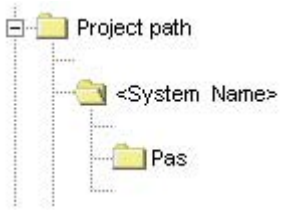
Differences between actions and functions

- 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 authoriation 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 809)
How to Use Actions From Other Sources (Page 823)
How to Rename an Action (Page 822)
How to Import an Action (Page 821)
How to Export an Action (Page 821)
How to Assign Authorizations (Page 820)
Triggers (Page 811)
How to Compile and Save an Action (Page 810)
Inserting Additional Action-Related Information (Page 807)
How to Edit Actions (Page 807)
How to Create a New Action (Page 806)
WinCC Coding Rule (Page 805)
How to Add Global Script Runtime to a Project's Startup List (Page 771)
Characteristics of Global Actions (Page 770)
Characteristics of Local Actions (Page 769)

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:

```
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 825)

WinCC Coding Rule (Page 805)

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 825)

How to Write Function Code (Page 794)

2.13.5 Inserting Additional 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.

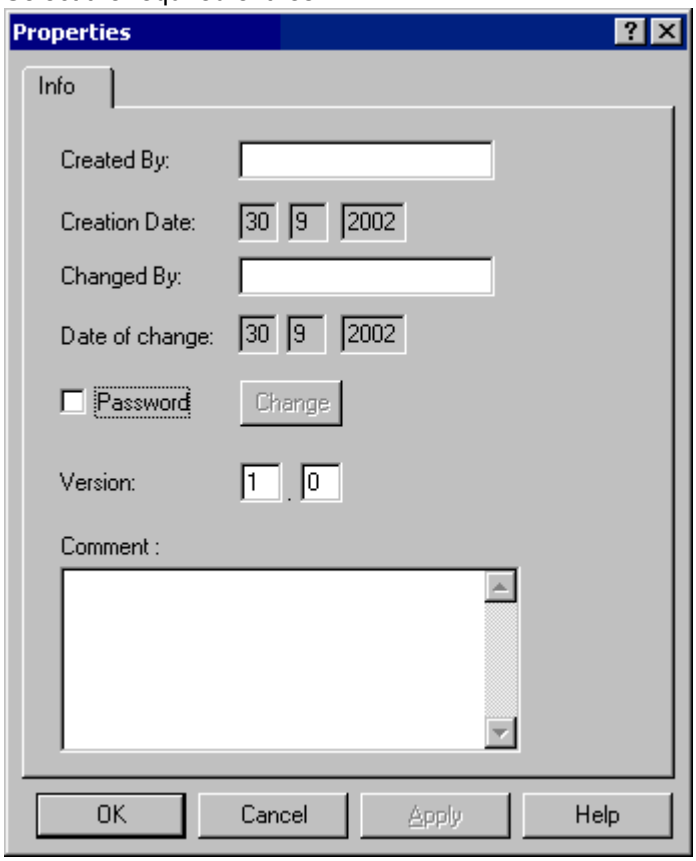
This dialog can be used to assign a password to protect the action against unauthorized read and write access.

Requirement

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 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 an Action Against Unauthorized Access (Page 809)

How to Set Different Views (Page 783)

Working with the Toolbars (Page 781)

2.13.6 How to Protect an Action Against Unauthorized Access

Introduction

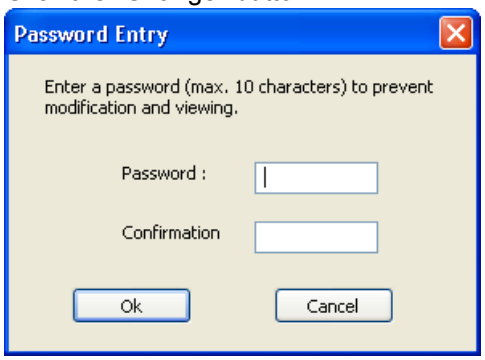
Actions can be protected with a password against unauthorized read and write access. The password is a part of the action-related information.

Requirement

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 the entries by clicking "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.

See also

How to Set Different Views (Page 783)

Working with the Toolbars (Page 781)

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. Click the button in the editing toolbar.
2. Examine the compiler messages in the lower portion of the edit window.
3. 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.
4. 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.
5. Click the button in the standard 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 824)

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.

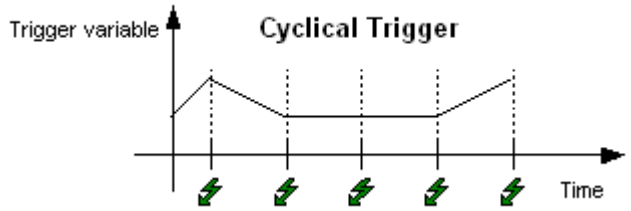
Tip: If the action is not to be carried out with each event occurrence, then a condition can be specified for the action that controls its further course dependent on the event. If the action is not to be executed any more, it can be terminated with a <value> return.

Rules for the Selection of Triggers

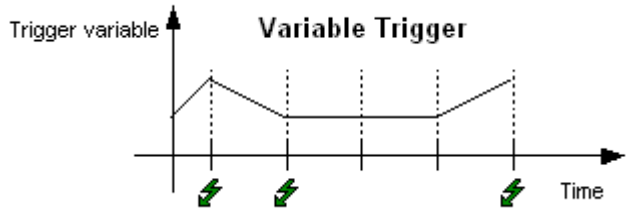
Depending on the system, it cannot be guaranteed that an action with a cyclic trigger is carried out at exactly the specified time. If this is a requirement, then the task (such as a check, etc.) must be implemented on the automation device.

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 (= action is performed).



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 818)

How to Change a Trigger (Page 817)

How to Add a New Trigger of the "Tag" Type (Page 815)

How to Add a New Trigger of the "Timer" Type (Page 814)

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 t his, 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.

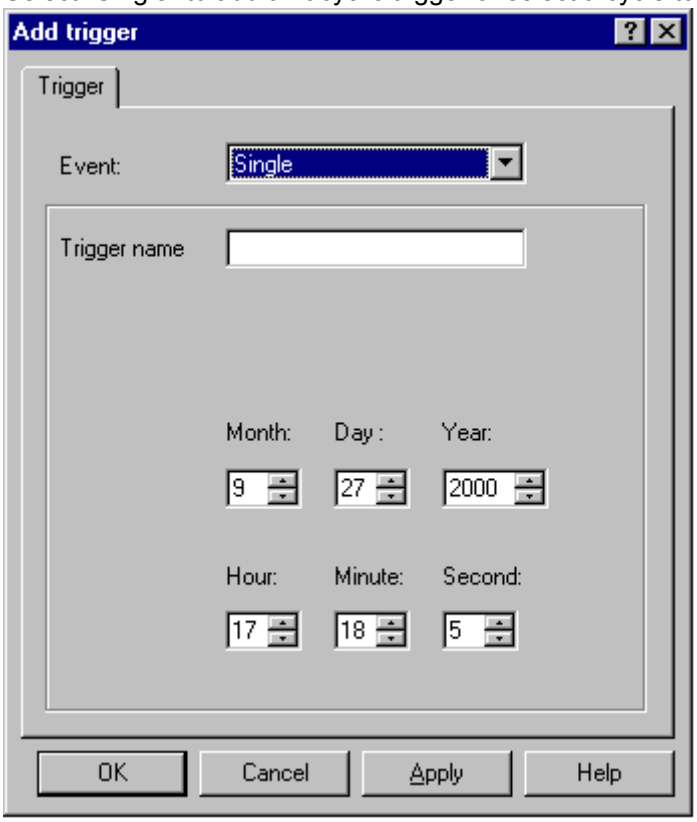
Requirement

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.

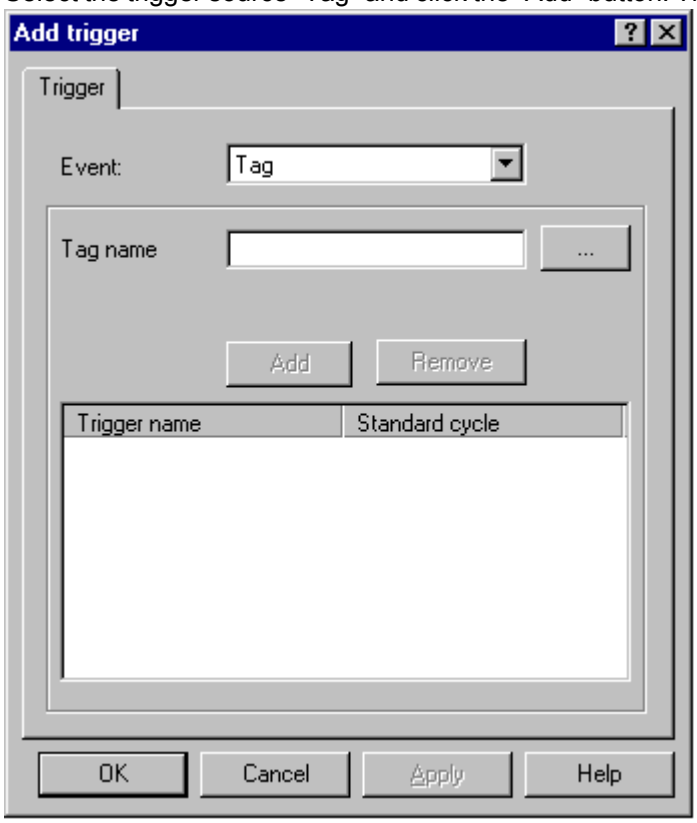
Depending on the choice of query method, it is possible that the tag changes while the system fails to detect this. In this case the action is not performed.

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 "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

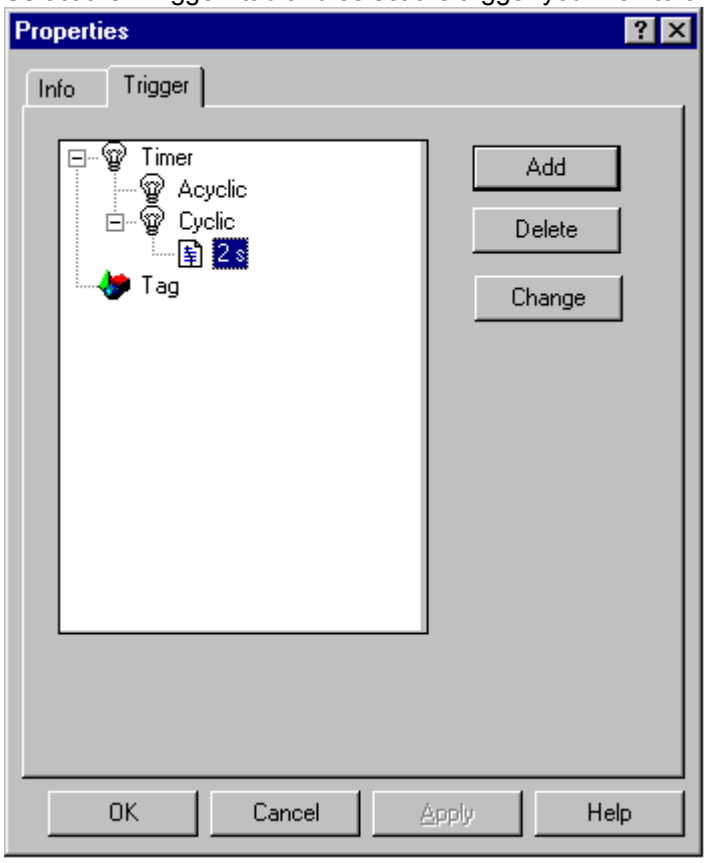
Triggers that have been defined can be changed at any time. They can also be changed in Runtime.

Requirement

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.

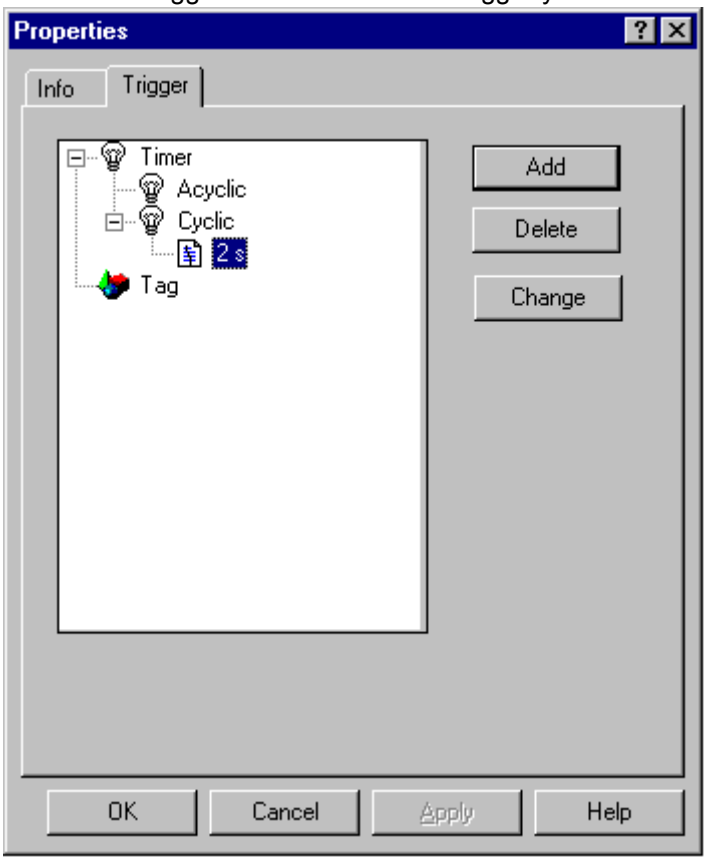
If a trigger is deleted in Runtime, it only takes effect after the action is saved.

Requirement

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

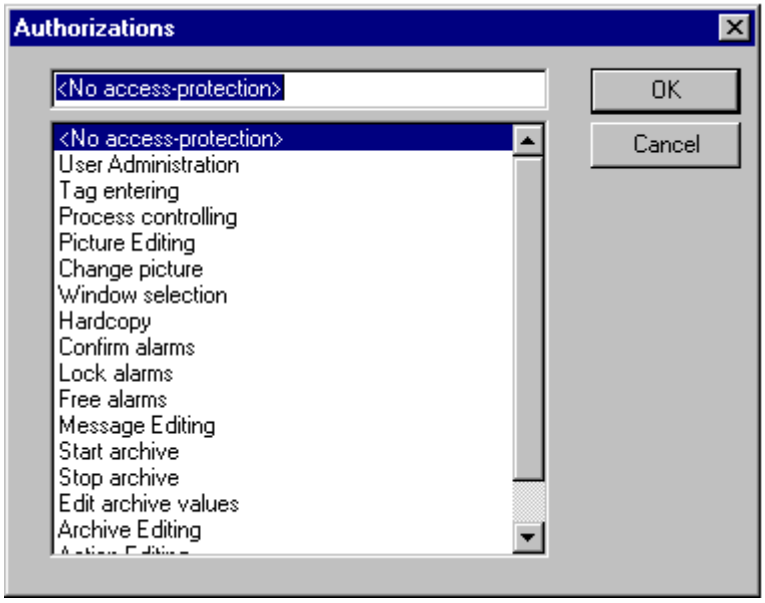
With the Global Script - Runtime diagnostic tool, you can influence the processing of actions in Runtime. Each action can be assigned an authorization. This authorization only effects the operation in the Global Script - Runtime window.

Requirement

The relevant action must be opened in the edit window.

Procedure

1. Select the "Operator authorization" command in the "Edit" menu.
   The "Authorizations" dialog box is opened.
2. Select an authorization.



3. Confirm your selection with "OK".

See also

GSC Runtime (Page 825)

2.13.10 How to Export an Action

Introduction

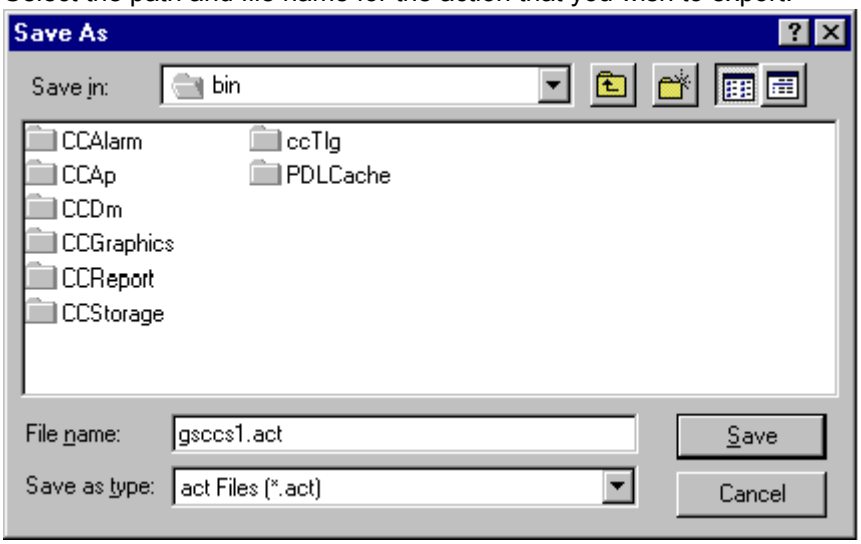
Use export and import to move actions between projects. The triggers linked to the actions are retained in the process.

Requirement

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

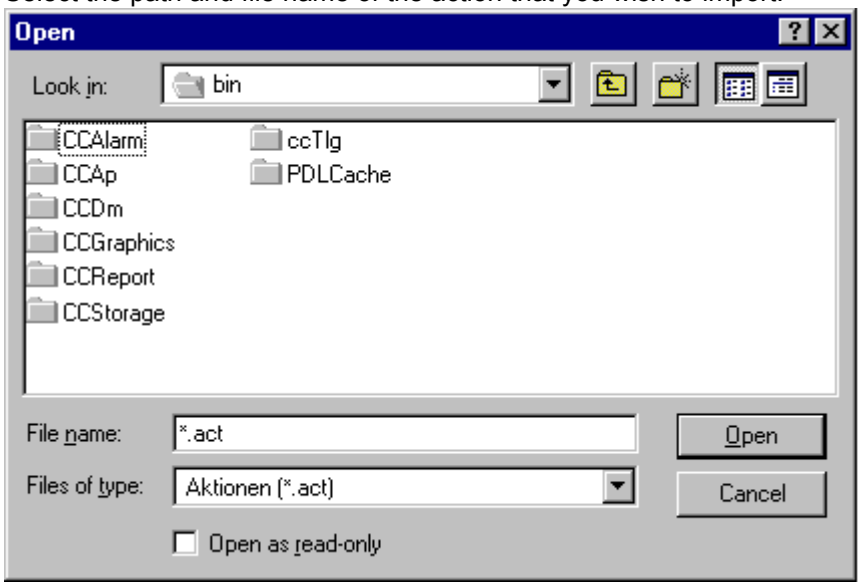
Introduction

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.

Requirement

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 821)

How to Export an Action (Page 821)

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.

See also

GSC Diagnose (Page 830)

GSC Runtime (Page 825)

2.14.2 GSC Runtime

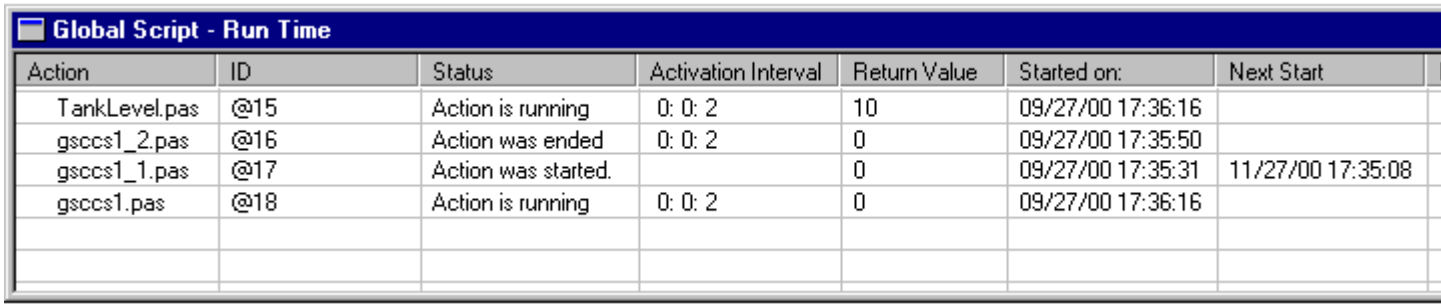
2.14.2.1 GSC Runtime

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:

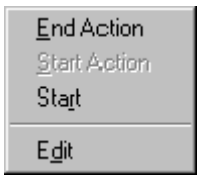
- No error occurred.
- The application is already connected to the script control. No additional connection setup is possible.
- There is no connection to the script control. Possibly no connection setup took place.
- An error occurred during interprocess communication. The cause of the error is unknown.
- 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 829)

Attributes of GSC Runtime (Page 829)

How to Place GSC Runtime in a Process Picture (Page 828)

How to Assign Authorizations (Page 820)

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.

See also

Attributes of GSC Runtime (Page 829)

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 802)

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.

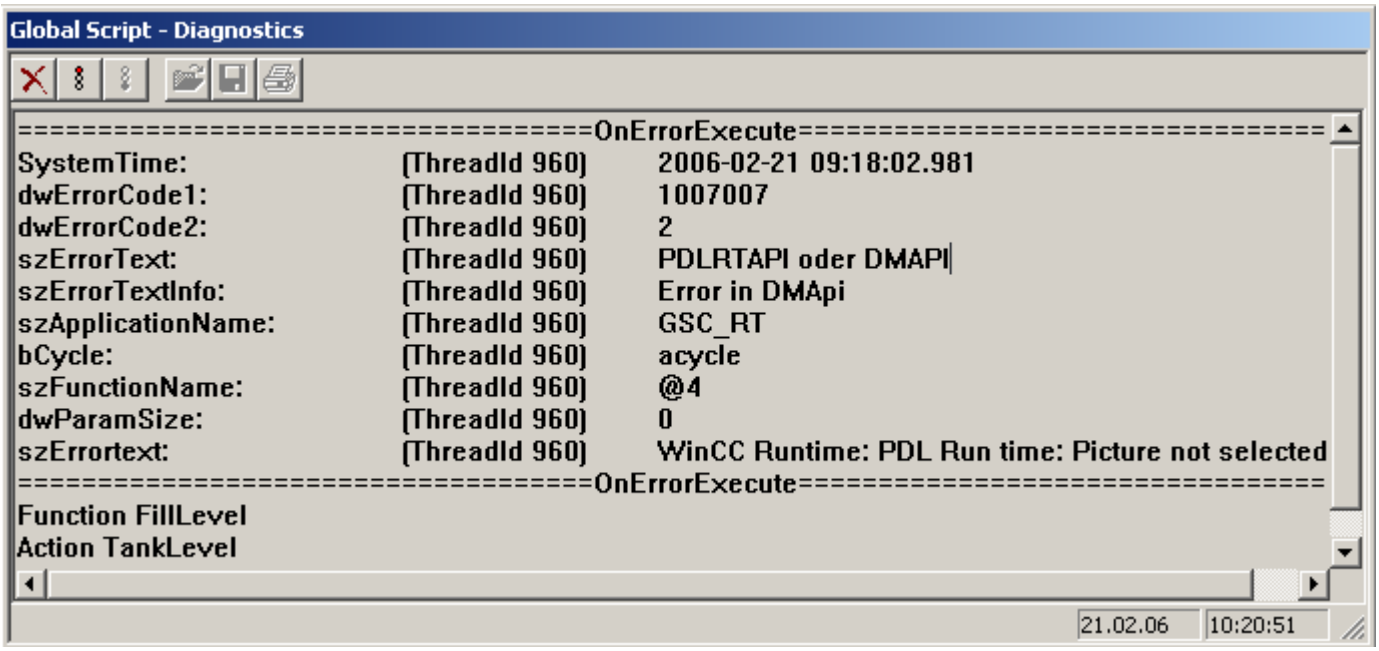


Figure 2-2 Diagnose window of GSC Diagnose

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 832)

Attributes of GSC Runtime (Page 829)

How to Place GSC Diagnose in a Process Picture? (Page 831)

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 829)

The Toolbar of GSC Diagnose (Page 832)

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:

Button	Function
	Deletes the contents of the window.
	Stops the updating of the window.
	Resumes the updating of the window.
	Opens a text file in the window.
	Saves the contents of the window in a text file.
	DPrints the contents of the window.

2.15 ANSI-C function descriptions

ANSI-C function descriptions 3

3 Resources

3.1 lpszPictureName

"lpszPictureName"

"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

You must not change the text in "lpszPictureName".

This would be possible using the "strcat()" function for example.

3.2 Standard functions

3.2.1 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)

3.2.2 Alarm

3.2.2.1 AcknowledgeMessage

Function

Acknowledges the message with the number that has been sent as a parameter in the message system.

Syntax

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.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

See also

AcknowledgeMessage example (Page 1471)

3.2.2.2 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.

See also

Structure definition CMN_ERROR (Page 1546)

Structure definition MSG_FILTER_STRUCT (Page 1552)

ResetFilter example (Page 1510)

AXC_SetFilter example (Page 1473)

3.2.2.3 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

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.

Name	Value	Description
FLAG_COMMENT_PARAMETER	0x00000001	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 "NULL."
FLAG_COMMENT_DIALOG	0x00000003	A comment dialog appears. The comment entered there is transferred to the message.
FLAG_TEXTID_PARAMETER	0x00000100	The text ID of a text from the TextLibrary is provided as the accompanying process value of the message.

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 proces 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

Value	Description
0	The function has been completed without any errors.
-101	The message editing could not be started.
-201	An error occurred when calling the "MSRTGetComment()" feature.
-301	An error occurred when calling the "MSRTCreateMsgInstanceWithComment()" feature.

3.2.2.4 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.

See also

Structure definition MSG_RTDATA_STRUCT (Page 1556)

3.2.3 Graphics

3.2.3.1 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.

3.2.3.2 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.

See also

GetLinkedVariable example (Page 1483)

3.2.3.3 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".

See also

GetLocalPicture example (Page 1483)

3.2.3.4 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

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

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.

See also

GetParentPicture example (Page 1485)

3.2.3.5 GetParentPictureWindow

Function

Provides a pointer to the name of the picture window.

Syntax

char* GetParentPictureWindow(char* lpszPictureName);

Parameters

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".

3.2.3.6 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.

Syntax

void OpenPicture(Picture PictureName)

Parameters

Picture name

Picture name

3.2.3.7 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

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 sting 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!

3.2.4 Report

3.2.4.1 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.

3.2.4.2 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.

See also

RPTJobPreview example (Page 1511)

3.2.4.3 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.

See also

RPTJobPrint example (Page 1511)

3.2.4.4 RptShowError

Function

This function provides an error decription 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.

STRUCTURES_TABLES_ERROR_STRUCTURE

3.2.5 WinCC

3.2.5.1 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.

3.2.5.2 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

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.

3.2.5.3 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.

3.2.5.4 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.

lpProperties

Name of the attrribute 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.

3.2.5.5 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

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 attrribute 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.

3.2.5.6 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()

3.2.5.7 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.

SystemTime	Time (UTC) at which the error occurred
dwErrorCode1	The error codes and their meaning are to be found in the structure definition
dwErrorCode2	The error codes and their meaning are to be found in the structure definition
szErrorText	Text description of the error cause
bCycle	Cycle type
szApplicationName	Error-triggering application
szFunctionName	FunctionID
szTagName	Tag name
dwCycle	Cycle type
szErrorTextTagName	Text description of the tag status
status	Tag status
lpszPictureName	Picture in which the error occurred
lpszObjectName	Object in which the error occurred
lpszPropertyName	Object property in which the error occurred
dwParamSize	only used internally
szErrorText	Text description of the error cause returned by the error structure "pError"

See also

Structure definition CCAPErrorExecute (Page 1544)

3.2.5.8 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

3.2.6 Windows

3.2.6.1 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.

3.2.6.2 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.

See also

ProgramExecute example (Page 1510)

3.2.7 Obsolete functions

3.2.7.1 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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

AXC_OnBtnEmergAckn

Function

External message window operation

This function opens the acknowledgement dialog (emergency acknowledgement/reset).

Syntax

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

AXC_OnBtnHornAckn

Function

External message window operation

This function acknowledges the horn signal.

Syntax

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.

FALSE

An error has occurred.

Note

The standard function is no longer supported for the new WinCC AlarmControl as of WinCC V7.0.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

AXC_OnBtnLockWin

Function

External message window operation.

This function calls the lock list.

Syntax

BOOL AXC_OnBtnLockWin (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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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 AlarmControl as of WinCC V7.0.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

Beispiel AXC_OnBtnMsgLast (Page 1472)

AXC_OnBtnMsgNext

Function

External message window operation

This function switches to the next message in the message list.

Syntax

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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 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.

See also

AXC_OnBtnScroll example (Page 1472)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

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.

See also

AXC_OnBtnSinglAckn example (Page 1472)

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.

See also

AXC_OnBtnMsgFirst example (Page 1471)

3.2.7.2 Report

ReportJob

Function

Depending on the value of the lpMethod Name parameter a print job or the preview for a print job is started.

Syntax

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.

3.2.7.3 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)

Parameters

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

All functions of this group have the same parameter and application.

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressHelpButton

Function

Displays the online help for the table window.

Syntax

BOOL TlgTableWindowPressHelpButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressInsertRecordButton

Syntax

BOOL TlgTableWindowPressInsertRecordButton(char* lpszWindowName)

Note

This function is no longer supported. A similar function exists in the User Archives runtime functions.

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressOpenArchiveVariableSelectionDlgButton

Function

Opens the dialog for connecting table columns to archives and tags.

Syntax

BOOL TlgTableWindowPressOpenArchiveVariableSelectionDlgButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressOpenDlgButton

Function

Opens the dialog for online configuration of the table window.

Syntax

BOOL TlgTableWindowPressOpenDlgButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressOpenDlgButton example (Page 1534)

TlgTableWindowPressOpenItemSelectDlgButton

Function

Opens the dialog for selecting the visible columns and the first column of the table window.

Syntax

BOOL TlgTableWindowPressOpenItemSelectDlgButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressOpenTimeSelectDlgButton

Function

Opens the dialog for setting the time range to be displayed in the table columns.

Syntax

BOOL TlgTableWindowPressOpenTimeSelectDlgButton(char* lpszWindowNumber)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTableWindowPressStartStopButton

Function

Updating of the table window is switched on or off (toggle function).

Syntax

BOOL TlgTableWindowPressStartStopButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTrendWindowPressHelpButton

Function

Displays the online help for the trend window.

Syntax

BOOL TlgTableWindowPressNextButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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

BOOL TlgTrendWindowPressLastButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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

BOOL TlgTableWindowPressNextButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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

BOOL TlgTrendWindowPressNextButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTrendWindowPressOneToOneButton

Function

Restores the standard size (1:1) in the trend window.

Syntax

BOOL TlgTrendWindowPressOneToOneButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTrendWindowPressOpenArchiveVariableSelectionDlgButton

Function

Opens the dialog for connecting trends to archives and tags.

Syntax

BOOL TlgTrendWindowPressOpenArchiveVariableSelectionDlgButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTrendWindowPressOpenDlgButton

Function

Opens the dialog for online configuration of the trend window.

Syntax

BOOL TlgTrendWindowPressOpenDlgButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

TlgTrendWindowPressOpenTimeSelectDlgButton

Function

Opens the dialog for setting the time range to be displayed.

Syntax

BOOL TlgTrendWindowPressOpenTimeSelectDlgButton(char* lpszWindowNumber)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

TlgTableWindowPressRemoveRecordButton

Syntax

BOOL TlgTableWindowPressRemoveRecordButton(char* lpszWindowName)

Note

This function is no longer supported. A similar function exists in the User Archives runtime functions.

TlgTrendWindowPressReportSaveButton

Function

The displayed Online Trend Control data is saved in a text file.

Syntax

BOOL TlgTrendWindowPressReportSaveButton (????)

Parameters

?????

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

TlgTrendWindowPressStartStopButton

Function

Updating of the trend window is switched on or off (toggle function).

Syntax

BOOL TlgTrendWindowPressStartStopButton(char* lpszWindowName)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressStartStopButton example (Page 1534)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressZoomInButton example (Page 1535)

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)

Parameters

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

All functions of this group have the same parameter and application.

See also

TlgTrendWindowPressZoomOutButton example (Page 1535)

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)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Table Control

Return value

Number of colums in a table window

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

int TlgGetNumberOfRows(char* lpszTemplate)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Table Control

Return value

Number of lines in the table window

See also

TlgGetNumberOfRows example (Page 1532)

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

int TlgGetNumberOfTrends(char* lpszTemplate)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Trend Control

Return value

Number of trends in the trend window

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

int TlgGetRowPosition(char* lpszTemplate)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Table Control

Return value

Current position of the line pointer in the table window

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

char* TlgGetRulerArchivNameTrend(char* lpszTemplate, int nTrend)

Parameters

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

See also

TlgGetRulerVariableNameTrend example (Page 1533)

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)

Parameters

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

See also

TlgGetRulerTimeTrend example (Page 1532)

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)

Parameters

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

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

char* TlgGetRulerVariableNameTrend(char* lpszTemplate, int nTrend)

Parameters

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

The tag name of the trend with the nTrend number in the trend window.

See also

TlgGetRulerVariableNameTrend example (Page 1533)

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

char* TlgGetTextAtPos(char* lpszTemplate, int nColumn, int nLine)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Table Control

nColumn

Number of the column

nLine

Number of the line

Return value

Content of the cell of a table window as text

See also

TlgGetRulerVariableNameTrend example (Page 1533)

TlgGetColumnPosition

Function

Provides the current position of the column pointer in the table window as column index.

Syntax

int TlgGetColumnPosition(char* lpszTemplate)

Parameters

lpszTemplate

Pointer to the window title of the WinCC Online Table Control

Return value

Current position of the column pointer in a table window

See also

TlgGetNumberOfColumns example (Page 1531)

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)

Parameters

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.

3.3 Internal functions

3.3.1 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 funtions 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

3.3.2 allocate

3.3.2.1 SysFree

Function

Releases the memory area previously reserved with the SysMalloc function.

Syntax

void SysFree(void* lpFree);

Parameters

lpFree

Pointer to the memory area reserved with the SysMalloc function

3.3.2.2 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

void* SysMalloc(unsigned long int size);

Parameters

size

Size of the memory area in bytes.

3.3.3 c_bib

3.3.3.1 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(), sprintf(), fprintf() can only process 360 characters in WinCC.

3.3.3.2 ctype

isahum

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.

isdigit

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.

isgraph

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.

3.3.3.3 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.

3.3.3.4 memory

memchr

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.

memcmp

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.

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.

memset

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.

3.3.3.5 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
- 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.

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.

fputs

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.

getc

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

fread

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:

- 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.

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.

ferror

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.

setvbuf

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.

tmpfile

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.

fseek

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.

fsetpos

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.

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.

Output

vfprintf

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.

vsprintf

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.

3.3.3.6 stdlib

abort

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.

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.

atoi

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.

div

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.

exit

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.

free

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.

ldiv

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.

strtoul

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.

system

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.

3.3.3.7 string

strcat

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.

strcpy

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.

strcspn

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.

strrchr

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.

strspn

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.

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.

3.3.3.8 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.

ctime

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.

3.3.4 graphics

3.3.4.1 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.

3.3.4.2 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 1490)

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 1490)

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);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

The main division lines on the graph scale are bold

FALSE

The main division lines on the graph scale are regular

See also

Beispiel GetScaling (Page 1490)

GetLongStrokesOnly

Function

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 1490)

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

lpszObjectName

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 1490)

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

STRUCTURES_TABLES_BAR_SCALING

See also

Bar Scaling (Page 1535)

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

Color chart (Page 1538)

GetBackColor example (Page 1474)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetBorderBackColor

Function

Specifies the background color of the lines or borders.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetBorderColor

Function

Specifies the line or border color as a numeric value.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetBorderColorBottom

Function

Specifies the 3D border color at the bottom.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorBottom

Function

When using slider objects, it specifies the color of the bottom limit.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetFillColor

Function

Specifies the color of the fill pattern.

Syntax

long int GetFillColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value of the fill color

Note

If the function is called in relation to the entire picture, set the parameter lpszObjectName = NULL.

See also

Color chart (Page 1538)

GetBackColor example (Page 1474)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)
Color chart (Page 1538)

GetScaleColor

Function

Specifies the scale color for bar objects.

Syntax

long int GetScaleColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value of the scale color for bar objects

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetSelBGColor

Function

Specifies the background color of the selected entry for the "text list" object.

Syntax

long int GetSelBGColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the background color of the selected entry

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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.

GetFilling

Function

Specifies whether dynamic filling with background color is activated.

Syntax

BOOL GetFilling(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1475)

GetFillingIndex

Function

Specifies the current fill level.

Syntax

long int GetFillingIndex(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Current fill level as a numeric value (0 - 100)

See also

GetFillingIndex example (Page 1476)

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.

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)

GetBackFlashColorOn

Function

Specifies the background flash color for the activated status.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)

Color chart (Page 1538)

GetBorderFlashColorOff

Function

Specifies the border or line flashing color for the deactivated status.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)
Color chart (Page 1538)

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 1477)

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 1477)

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 1477)

GetFlashRateBackColor

Function

Specifies the flash frequency of the background.

Syntax

long int GetFlashRateBackColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Flash frequency of the background

STRUCTURES_TABLES_FLASH_FREQUENCIES

See also

GetFlashBackColorOn example (Page 1478)

Flash frequencies (Page 1536)

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

STRUCTURES_TABLES_FLASH_FREQUENCIES

See also

GetFlashBackColorOn example (Page 1478)

Flash frequencies (Page 1536)

GetFlashRateForeColor

Function

Specifies the flash frequency of the font.

Syntax

long int GetFlashRateForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Flash frequency of the font

STRUCTURES_TABLES_FLASH_FREQUENCIES

See also

GetFlashBackColorOn example (Page 1478)

Flash frequencies (Page 1536)

GetForeFlashColorOff

Function

Specifies the font flash color for the deactivated status.

Syntax

long int GetForeFlashColorOff(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Font flash color for the deactivated status as a numeric value

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)

Color chart (Page 1538)

GetForeFlashColorOn

Function

Specifies the font flash color for the activated status.

Syntax

long int GetForeFlashColorOn(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Font flash color for the activated status as a numeric value

STRUCTURES_TABLES_COLOR_CHART

See also

GetFlashBackColorOn example (Page 1478)

Color chart (Page 1538)

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.

See also

GetFocus example (Page 1479)

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.

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

STRUCTURES_TABLES_TEXT_ALIGNMENT

See also

Text alignment (Page 1543)

GetFontSize example (Page 1480)

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

STRUCTURES_TABLES_TEXT_ALIGNMENT

See also

GetFontSize example (Page 1480)

Text alignment (Page 1543)

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 1479)

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 1479)

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 seleted

See also

GetText example (Page 1507)

GetFontSize

Function

Specifies the font size.

Syntax

long int GetFontSize(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Current font size

See also

GetFontSize example (Page 1480)

GetFontUnderline

Function

Specifies whether the font is underlined or not.

Syntax

BOOL GetFontUnderline(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Underlined font on

FALSE

Underlined font off

See also

GetFontBold example (Page 1479)

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 1479)

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

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 1507)

general

GetLayer

Function

Specifies the picture layer in which the object is located.

Syntax

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

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 1482)

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 1508)

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

long int GetDirection(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Bar direction of bar objects as numeric value

STRUCTURES_TABLES_BAR_DIRECTION

See also

Bar direction (Page 1535)

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 1480)

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 1482)

GetPointCount

Function

Specifies the number of corners of a polygon or in a polygon line.

Syntax

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

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 1480)

GetRadiusHeight

Function

Specifies the radius of an ellipse, ellipse segment or elliptical arc in a vertical direction.

Syntax

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 1480)

GetRadiusWidth

Function

Specifies the radius of an ellipse, ellipse segment or elliptical arc in a horizontal direction.

Syntax

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 1480)

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

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 1482)

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

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 1508)

GetRotationAngle

Function

Specifies the angle of rotation about the central axis for lines, polygons and polylines.

Syntax

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 1480)

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 1480)

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 1509)

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 1480)

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 1508)

GetWidth

Function

Specifies the width of the rectangle framing an object.

Syntax

long int GetWidth(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 1509)

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 1480)

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.

STRUCTURES_TABLES_LIST_TYPES

See also

List types (Page 1541)

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

lpszObjectName

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
STRUCTURES_TABLES_LIST_TYPES

See also

GetHiddenInput example (Page 1481)
List types (Page 1541)

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.

FALSE

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

STRUCTURES_TABLES_IO_FIELD_DATA_TYPE_OF_FIELD_CONTENT

See also

GetHiddenInput example (Page 1481)

I/O field, data type of the field content (Page 1537)

GetHiddenInput

Function

Specifies whether hidden input is activated for I/O fields.

Syntax

BOOL GetHiddenInput(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Hidden input is activated

FALSE

Hidden input is not activated

See also

GetHiddenInput example (Page 1481)

GetInputValueChar

Function

Specifies the input value in the data type "char" for I/O fields.

Syntax

```
char* GetInputValueChar(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);
```

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Pointer to the input value in the data type "char"

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

STRUCTURES_TABLES_LIST_TYPES

See also

GetHiddenInput example (Page 1481)

List types (Page 1541)

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 1481)

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

STRUCTURES_TABLES_IA_FIELD_OUTPUT_FORMAT

See also

I/O field, data type of the field content (Page 1537)

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

lpszObjectName

Object name

Return value

Pointer to the output value in the data type "char"

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

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 1485)

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 1474)

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 1474)

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 1484)

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 1484)

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 limt value reserve 4 is monitored.

FALSE

In case of bar objects the upper limt value reserve 4 is not monitored.

See also

GetMarker example (Page 1484)

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 limt value reserve 5 is monitored.

FALSE

In case of bar objects the upper limt value reserve 5 is not monitored.

See also

GetMarker example (Page 1484)

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 limt value reserve 4 is monitored.

FALSE

In case of bar objects the lower limt value reserve 4 is not monitored.

See also

GetMarker example (Page 1484)

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 limt value reserve 5 is monitored.

FALSE

In case of bar objects the lower limt value reserve 5 is not monitored.

See also

GetMarker example (Page 1484)

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 1484)

GetCheckToleranceLow

Function

When using bar objects, it specifies whether the lower tolerance limit is monitored.

Syntax

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 1484)

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 1484)

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

lpszObjectName

Object name

Return value

TRUE

In case of bar objects the lower warning limit is monitored.

FALSE

In case of bar objects the lower warning limit is not monitored.

See also

GetMarker example (Page 1484)

GetColorAlarmHigh

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorAlarmLow

Function

Specifies the bar color for bar objects upon reaching the lower alarm limit.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorLimitHigh4

Function

Specifies the bar color for bar objects upon reaching the upper limit reserve 4.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

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

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the bar color upon reaching the lower limit reserve 4

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorLimitLow5

Function

Specifies the bar color for bar objects upon reaching the lower limit reserve 5.

Syntax

long int GetColorLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the bar color upon reaching the lower limit reserve 5

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorToleranceHigh

Function

Specifies the bar color for bar objects upon reaching the upper tolerance limit.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorToleranceLow

Function

Specifies the bar color for bar objects upon reaching the lower tolerance limit.

Syntax

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

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorWarningHigh

Function

Specifies the bar color for bar objects upon reaching the upper warning limit limit.

Syntax

long int GetColorWarningHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the bar color upon reaching the upper warning limit

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetColorWarningLow

Function

Specifies the bar color for bar objects upon reaching the lower warning limit.

Syntax

long int GetColorWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the bar color upon reaching the lower warning limit

STRUCTURES_TABLES_COLOR_CHART

See also

GetBackColor example (Page 1474)

Color chart (Page 1538)

GetLimitHigh4

Function

Specifies the upper limit value for reserve 4 for bar objects.

Syntax

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 1474)

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 1474)

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 1474)

GetLimitLow5

Function

Specifies the low limit value for reserve 5 for bar objects.

Syntax

double GetLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Low limit value for reserve 5 for bar objects

See also

GetAlarmHigh example (Page 1474)

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 1474)

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 1474)

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 1484)

GetToleranceHigh

Function

Specifies the upper tolerance limit for bar objects.

Syntax

double GetToleranceHigh(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Upper tolerance limit for bar objects

See also

GetAlarmHigh example (Page 1474)

GetToleranceLow

Function

Specifies the lower tolerance limit for bar objects.

Syntax

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 1474)

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 1484)

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 1484)

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 1484)

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 1484)

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 1484)

GetTypeLimitLow5

Function

Specifies for bar objects whether the lower limit reserve 5 is given in percentages or absolute terms.

Syntax

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 1484)

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 1484)

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 1484)

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 1484)

GetTypeWarningLow

Function

Specifies for bar objects whether the lower warning limit is given in percentages or absolute terms.

Syntax

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 1484)

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 1474)

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 1474)

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 1551)

GetLink example (Page 1482)

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 1508)

GetAdaptPicture

Function

Specifies for picture windows whether the picture is to be adapted to the window size.

Syntax

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 1508)

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 1508)

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 1508)

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

STRUCTURES_TABLES_IO_FIELD_TYPE

See also

I/O field, field type (Page 1537)

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 1508)

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

lpszObjectName

Object name

Return value

TRUE

Picture window can be closed

FALSE

Picture window cannot be closed

See also

GetVisible example (Page 1508)

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 sagment

FALSE

Color change applies to the entire bar

See also

GetVisible example (Page 1508)

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 1508)

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

lpszPictureName

Picture name

lpszObjectName

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 1508)

GetEditAtOnce

Function

Specifies whether the "Immediate input" property is activated for I/O fields.

Syntax

BOOL GetEditAtOnce(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

"Immediate input" property is activated

FALSE

"Immediate input" property is deactivated

See also

GetVisible example (Page 1508)

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 1508)

GetHotkey

Function

Specifies the key combination for check boxes.

Syntax

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

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 1508)

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 fot 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 1508)

GetLastChange

Function

Specifies the date when the picture was last changed.

Syntax

char* GetLastChange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Date of the last change of the picture

See also

GetPictureName example (Page 1486)

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 1508)

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

lpszObjectName

Object name

Return value

TRUE

Picture or application window is movable

FALSE

Picture or application window is not movable

See also

GetVisible example (Page 1508)

GetOffsetLeft

Function

Specifies the horizontal picture distance from the left window border for picture windows.

Syntax

long int GetOffsetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Horizontal picture distance from the left window border for picture windows

GetOffsetTop

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 1508)

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 1508)

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 1508)

GetOperationReport

Function

Specifies for all objects except application and picture windows and OLE control whether the reason for the operation is logged.

Syntax

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 1508)

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 1486)

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

lpszPictureName

Picture name

lpszObjectName

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 corrsponds 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 lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Picture window has a scroll bar

FALSE

Picture window has no scroll bar

See also

GetVisible example (Page 1508)

GetServerName

Function

Specifies the default setting for the process value to be displayed for OLE control and OLE object.

Syntax

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.

See also

GetPictureName example (Page 1486)

GetSizeable

Function

Specifies for application or picture windows whether the window size can be changed.

Syntax

BOOL GetSizeable(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Application or picture window is sizeable

FALSE

Application or picture window is not sizeable

See also

GetVisible example (Page 1508)

GetSmallChange

Function

Specifies the number of steps for slider objects by which the slider is shifted by a mouse click.

Syntax

long int GetSmallChange(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

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

See also

GetTagPrefix example (Page 1501)

GetTrend

Function

When using bar objects, it specifies whether the trend display is activated.

Syntax

BOOL GetTrend(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Trend display is activated for a bar object

FALSE

Trend display is not activated for a bar object

See also

GetVisible example (Page 1508)

GetUpdateCycle

Function

Specifies the update cycle for the entire picture.

Syntax

long int GetUpdateCycle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Numeric value defining the update cycle

See also

Structure definition LINKINFO (Page 1551)

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 1508)

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 1508)

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 windos.

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 1488)

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 1489)

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 1490)

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

Return 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 1538)

GetBackColor example (Page 1474)

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 1538)

GetPictureDown example (Page 1486)

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 1538)
GetBackColor example (Page 1474)

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

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

Note

Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also

GetPictureDown example (Page 1486)

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

Note

Bitmap files (*.bmp, *.dib) as well as metafiles (*.emf, *.wmf) can be integrated.

See also

GetPictureUp example (Page 1487)

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

lpszObjectName

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 lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1538)

GetBackColor example (Page 1474)

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

lpszObjectName

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

lpszPictureName

Picture name

lpszObjectName

Object name

lpszPropertyName

Name of the object property

Return value

Value of the attribute in the data type "BOOL"

See also

GetPropBOOL example (Page 1488)

GetPropChar

Function

Specifies the current status of a property of the data type "char".

Syntax

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

See also

GetPropChar example (Page 1489)

GetPropDouble

Function

Specifies the current status of a property of the data type "double".

Syntax

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

long GetPropWord(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 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 lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

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

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 1538)
GetBackColor example (Page 1474)

GetBasePicture

Function

Specifies the basic picture name fur the status display.

Syntax

char* GetBasePicture(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Basic picture name for the status display

GetBasePicUseTransColor

Function

When using the status display, it specifies whether the transparent color is used for the basic picture.

Syntax

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

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

The flash picture was not stored in the object.

FALSE

The flash picture was stored in the object.

GetFlashPicTransColor

Function

Specifies the transparent color of the flash picture for the status display.

Syntax

long int GetFlashPicTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Transparent color of the flash picture as numeric value

Note

This function only applies to Bitmap graphics (*.bmp).

See also

Color chart (Page 1538)

GetBackColor example (Page 1474)

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)

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

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 1536)

GetFlashRateFlashPic example (Page 1478)

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

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

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 1475)

GetBorderEndStyle

Function

Specifies the type of line end.

Syntax

long int GetBorderEndStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Type of line end as numeric value

STRUCTURES_TABLES_LINE_ENDS

See also

GetBorderStyle example (Page 1475)

Line end style (Page 1541)

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

STRUCTURES_TABLES_LINE_STYLES

See also

GetBorderStyle example (Page 1475)

Line styles (Page 1541)

GetBorderWidth

Function

Specifies the line or border line width.

Syntax

long int GetBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Line or border line width as numeric value

See also

GetBorderStyle example (Page 1475)

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

STRUCTURES_TABLES_TEXT_ALIGNMENT

See also

GetBorderStyle example (Page 1475)

Text alignment (Page 1543)

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 1539)

GetFillStyle example (Page 1476)

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 1539)

GetFillStyle example (Page 1476)

GetItemBorderStyle

Function

Specifies the dividing line style for the "text list" object.

Syntax

long int GetItemBorderStyle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

Dividing line style for the "text list" object

STRUCTURES_TABLES_LINE_STYLES

See also

GetBorderStyle example (Page 1475)

Line styles (Page 1541)

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 1475)

GetPressed

Function

Specifies for 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 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

lpszPictureName

Picture name

lpszObjectName

Object name

Return value

TRUE

Button is displayed in the usual Windows fashion.

FALSE

The apperance of the button is defined by the user.

3.3.4.3 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

BOOL SetAlignment(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAlignment);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1522)

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

lpszPictureName

Picture name

lpszObjectName

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 1522)

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 1522)

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 1522)

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 1522)

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 1522)

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

lpszObjectName

Object name

lLongStrokesSize

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 1522)

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

lpszPictureName

Picture name

lpszObjectName

Object name

lRightComma

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 1522)

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 1522)

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 1522)

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 1535)

SetScaling example (Page 1522)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetBackColorBottom

Function

Sets the background color of the slider objects at the bottom right.

Syntax

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

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

Color chart (Page 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetBorderBackColor

Function

Sets the background color of the lines or borders.

Syntax

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetBorderColorBottom

Function

Sets the 3D border color at the bottom.

Syntax

BOOL SetBorderColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColorBottom);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lBorderColorBottom

Numeric value defining the 3-D border color at the bottom

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

Color chart (Page 1538)

SetBackColor example (Page 1512)

SetBorderColorTop

Function

Sets the 3D border color at the top.

Syntax

BOOL SetBorderColorTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColorTop);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetColorBottom

Function

When using slider objects, it sets the color of the bottom limit.

Syntax

BOOL SetColorBottom(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorBottom);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lColorBottom

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetFillColor

Function

Sets the color of the fill pattern.

Syntax

BOOL SetFillColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillColor);

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 1538)

SetBackColor example (Page 1512)

SetForeColor

Function

Sets the font color.

Syntax

BOOL SetForeColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeColor);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lForeColor

Numeric value defining the font color

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

Color chart (Page 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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.

See also

Color chart (Page 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetTrendColor

Function

Sets the trend color for bar objects.

Syntax

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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 1514)

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 1514)

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 1538)

SetFlashBackColorOn example (Page 1515)

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 1538)

SetFlashBackColorOn example (Page 1515)

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 1515)

Color chart (Page 1538)

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 1538)

SetFlashBackColorOn example (Page 1515)

SetFlashBackColor

Function

Activates or deactivates background flashing.

Syntax

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 1515)

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 1515)

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 1515)

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 1536)

SetFlashBackColor example (Page 1515)

SetFlashRateBorderColor

Function

Sets the flash frequency of the line or border.

Syntax

BOOL SetFlashRateBorderColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateBorderColor);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lFlashRateBorderColor

Flash frequency of the line or border

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

Flash frequencies (Page 1536)

SetFlashBackColor example (Page 1515)

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 1536)

SetFlashBackColor example (Page 1515)

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 1538)

SetFlashBackColorOn example (Page 1515)

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 1538)

SetFlashBackColorOn example (Page 1515)

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.

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 1516)

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 1543)

SetFontSize example (Page 1516)

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 1543)

SetFontSize example (Page 1516)

SetFontBold

Function

Switches the bold font on or off.

Syntax

BOOL SetFontBold(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontBold);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1516)

SetFontItalic

Function

Switches the italic font on or off.

Syntax

BOOL SetFontItalic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontItalic);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bFontItalic

Italic font on/off

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetFontBold example (Page 1516)

SetFontName

Function

Sets a font.

Syntax

BOOL SetFontName(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szFontName);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

szFontName

Pointer to name of font

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetText example (Page 1529)

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 1516)

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 1516)

SetOrientation

Function

Defines the text orientation (vertical/horizontal).

Syntax

BOOL SetOrientation(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOrientation);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1516)

SetText

Function

Sets the value of the "text" property for objects like static text, check box or radio box.

Syntax

BOOL SetText(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szText);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1529)

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.

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 1517)

SetActualPointTop

Function

Sets the Y value for the current point of a polygon or polygon line.

Syntax

BOOL SetActualPointTop(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lActualPointTop);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lActualPointTop

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 1529)

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

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 1535)

SetTop example (Page 1529)

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 1529)

SetHeight

Function

Sets the height of the rectangle framing an object.

Syntax

BOOL SetHeight(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lHeight);

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 1516)

SetLeft

Function

Sets the X value of the upper left corner of the rectangle framing an object

Syntax

BOOL SetLeft(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLeft);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lLeft

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 1517)

SetPointCount

Function

Sets the number of corners of a polygon or in a polygon line.

Syntax

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 1517)

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 1516)

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 1516)

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

lpszObjectName

Object name

lRadiusWidth

Radius in horizontal direction

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetWidth example (Page 1530)

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 lReferenceRotationLeft);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lReferenceRotationLeft

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 1517)

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 1529)

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 1517)

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 1516)

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

lpszObjectName

Object name

lRoundCornerWidth

Horizontal radius

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetWidth example (Page 1530)

SetStartAngle

Function

Sets the start angle of circle and ellipse segments and circle and elliptical arcs.

Syntax

BOOL SetStartAngle(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lStartAngle);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lStartAngle

Starting angle

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetHeight example (Page 1516)

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 lTop);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lTop

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

SetTop example (Page 1529)

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 1530)

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 1529)

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 1517)

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 1517)

SetBitNumber

Function

Sets the relevant bit in the output value for the "bit" list type.

Syntax

BOOL SetBitNumber(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBitNumber);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lBitNumber

Relevant bit in the output value for the "bit" list type

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

List types (Page 1541)

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 1517)

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 1517)

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 1517)

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 1519)

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 1511)

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 1511)

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 1518)

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

lpszObjectName

Object name

bCheckAlarmLow

Monitoring yes/no

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetMarker example (Page 1518)

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 1518)

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 1518)

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 1518)

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

bCheckLimitLow5

Monitoring yes/no

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetMarker example (Page 1518)

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 1518)

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 1518)

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 1518)

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 1518)

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 1538)

SetBackColor example (Page 1512)

SetColorAlarmLow

Function

Sets the bar color for bar objects upon reaching the lower alarm limit.

Syntax

BOOL SetColorAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorAlarmLow);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lColorAlarmLow

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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

lColorLimitHigh5

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetColorLimitLow5

Function

Sets the bar color for bar objects upon reaching the lower limit reserve 5.

Syntax

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 1538)

SetBackColor example (Page 1512)

SetColorToleranceHigh

Function

Sets the bar color for bar objects upon reaching the upper tolerance limit.

Syntax

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 1538)

SetBackColor example (Page 1512)

SetColorToleranceLow

Function

Sets the bar color for bar objects upon reaching the lower tolerance limit.

Syntax

BOOL SetColorToleranceLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorToleranceLow);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lColorToleranceLow

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

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

lColorWarningLow

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 1538)

SetBackColor example (Page 1512)

SetLimitHigh4

Function

Sets the high limit value for reserve 4 for bar objects.

Syntax

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 1511)

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 1511)

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 1511)

SetLimitLow5

Function

Sets the low limit value for reserve 5 for bar objects.

Syntax

BOOL SetLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitLow5);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

dLimitLow5

Low 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 1511)

SetLimitMax

Function

Sets the high limit value for I/O fields.

Syntax

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 1511)

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 1511)

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 1518)

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 1511)

SetToleranceLow

Function

Sets the lower tolerance limit for bar objects.

Syntax

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 1511)

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 1518)

SetTypeAlarmLow

Function

Specifies for bar objects whether the lower alarm limit is given in percentages or absolute terms.

Syntax

BOOL SetTypeAlarmLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeAlarmLow);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bTypeAlarmLow

Lower 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 1518)

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 1518)

SetTypeLimitHigh5

Function

Specifies for bar objects whether the upper limit for reserve 5 is given in percentages or absolute terms.

Syntax

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 1518)

SetTypeLimitLow4

Function

Specifies for bar objects whether the lower limit for reserve 4 is given in percentages or absolute terms.

Syntax

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 1518)

SetTypeLimitLow5

Function

Specifies for bar objects whether the lower limit for reserve 5 is given in percentages or absolute terms.

Syntax

BOOL SetTypeLimitLow5(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitLow5);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bTypeLimitLow5

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 1518)

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 1518)

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 1518)

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 1518)

SetTypeWarningLow

Function

Specifies for bar objects whether the lower warning limit is given in percentages or absolute terms.

Syntax

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 1518)

SetWarningHigh

Function

Sets the upper warning limit for bar objects.

Syntax

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 1518)

SetWarningLow

Function

Sets the lower warning limit for bar objects.

Syntax

BOOL SetWarningLow(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dWarningLow);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

dWarningLow

Lower warning limit

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetMarker 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.

SetLink

Function

Creating a tag connection of object properties

Syntax

BOOL SetLink(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, LPLINKINFO *pLink);

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 1551)

SetLink example (Page 1518)

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 1530)

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

lpszObjectName

Object name

lBoxType

Field type

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

I/O field, field type (Page 1537)

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

SetVisible example (Page 1530)

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.

See also

SetVisible example (Page 1530)

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
FALSE	Alpha-Cursor

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

Example - SetCursorMode (Page 1513)

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 1530)

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 1530)

SetHysteresis

Function

When using bar objects, it specifies whether the display appears with or without hysteresis.

Syntax

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 1530)

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

lpszObjectName

Object name

dMin

Minimum value

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

SetOffsetLeft

Function

Sets the horizontal picture distance from the left window border for picture windows.

Syntax

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 1530)

SetOperationMessage

Function

Controls the output of a message when operating the objects "I/O field", "Check box", "Radio box" and "Slider".

Syntax

BOOL SetOperationMessage(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperationMessage);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bOperationMessage

Message output for yes/no operation

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetVisible example (Page 1530)

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 1530)

SetPasswordLevel

Function

Defines the authorization level for operating objects for all objects except application and picture windows and OLE control.

Syntax

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

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 1519)

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 corrsponds 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

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 1526)

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 1530)

SetVisible

Function

Controls the display of an object.

Syntax

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 1530)

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

BOOL SetPosition(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPosition);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lPosition

Slider position of the OCX slider object

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

See also

SetPosition example (Page 1520)

SetRangeMax

Function

Defines the adjustment range "Max" of the OCX slider object.

Syntax

BOOL SetRangeMax(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRangeMax);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

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 1521)

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 1521)

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetPicDownUseTransColor

Function

Controls the transparent color for the "On/pressed" status of a round button.

Syntax

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

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 1538)

SetBackColor example (Page 1512)

SetPictureDeactivated

Function

Specifies the picture name for the "deactivated" status of a round button.

Syntax

BOOL SetPictureDeactivated(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureDeactivated);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

szPictureDeactivated

Picture name for "deactivated" 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 1519)

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 1519)

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 1520)

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 lPicUpTransparent);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lPicUpTransparent

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 1538)

SetBackColor example (Page 1512)

SetPicUpUseTransColor

Function

Controls the transparent color for the "Off/not pressed" status of a round button.

Syntax

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

BOOL SetPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicUseTransColor);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bPicUseTransColor

Transparent color yes/no

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 1520)

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 1489)

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 "lValue".

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

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

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 1538)

SetBackColor example (Page 1512)

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 1538)

SetBackColor example (Page 1512)

SetFlashPicUseTransColor

Function

Controls the transparent color of the flash picture for a status display.

Syntax

BOOL SetFlashPicUseTransColor(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashPicUseTransColor);

Parameters

lpszPictureName

Picture name

lpszObjectName

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

BOOL SetFlashRateFlashPic(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateFlashPic);

Parameters

lpszPictureName

Picture name

lpszObjectName

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 1536)

SetFlashRateFlashPic example (Page 1515)

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

lIndex

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 lBackBorderWidth);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lBackBorderWidth

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 1513)

SetBorderEndStyle

Function

Sets the type of line end.

Syntax

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 1541)

SetBorderEndStyle example (Page 1512)

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 1541)

SetBorderStyle example (Page 1513)

SetBorderWidth

Function

Sets the line or border line width.

Syntax

BOOL SetBorderWidth(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderWidth);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lBorderWidth

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 1513)

SetBoxAlignment

Function

Defines the arrangement of controls (left or right justified) in check boxes or radio boxes.

Syntax

BOOL SetBoxAlignment(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxAlignment);

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

lBoxAlignment

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 1538)

SetBorderStyle example (Page 1513)

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

lFillStyle

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 1539)

SetFillStyle example (Page 1514)

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

lFillStyle2

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 1539)

SetFillStyle example (Page 1514)

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 1541)

SetBorderStyle example (Page 1513)

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 1513)

SetPressed

Function

Specifies for round buttons whether the switch setting is "pressed" or "not pressed".

Syntax

```
BOOL SetPressed(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL
bPressed);
```

Parameters

lpszPictureName

Picture name

lpszObjectName

Object name

bPressed

Switch setting of the round 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 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

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

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

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

BOOL StorePicture();

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

3.3.5 tag

3.3.5.1 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.

3.3.5.2 get

Functionality of the GetTag functions

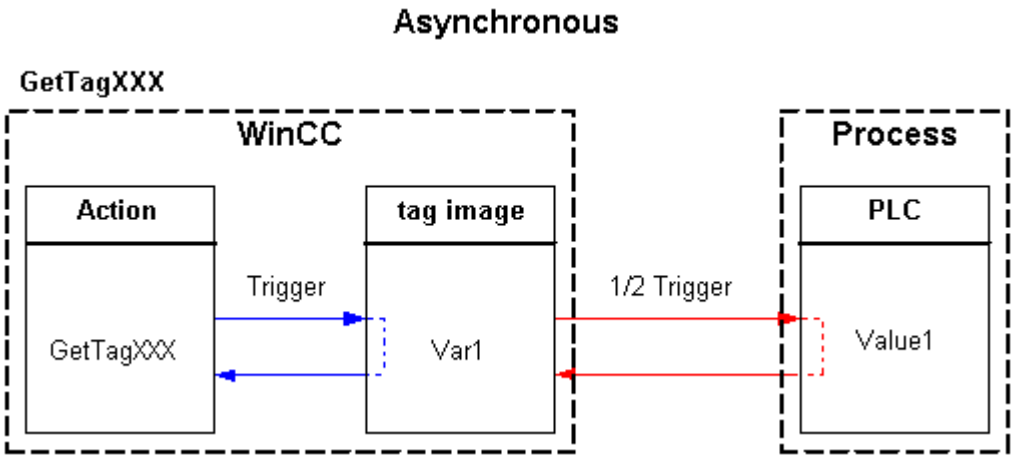
GetTagXXX

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



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.

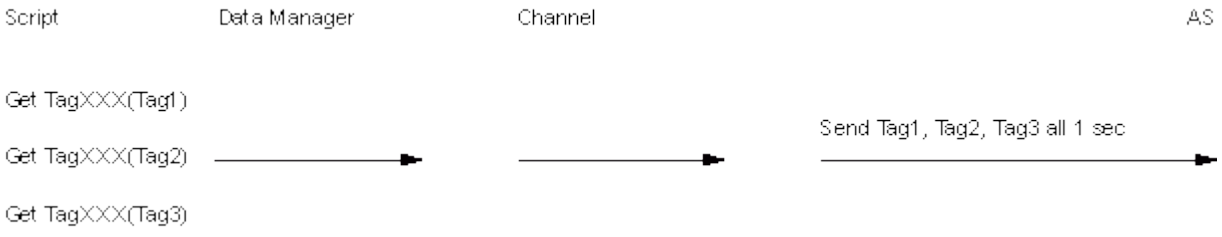
Behavior in actions with tag trigger (recommended):

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.

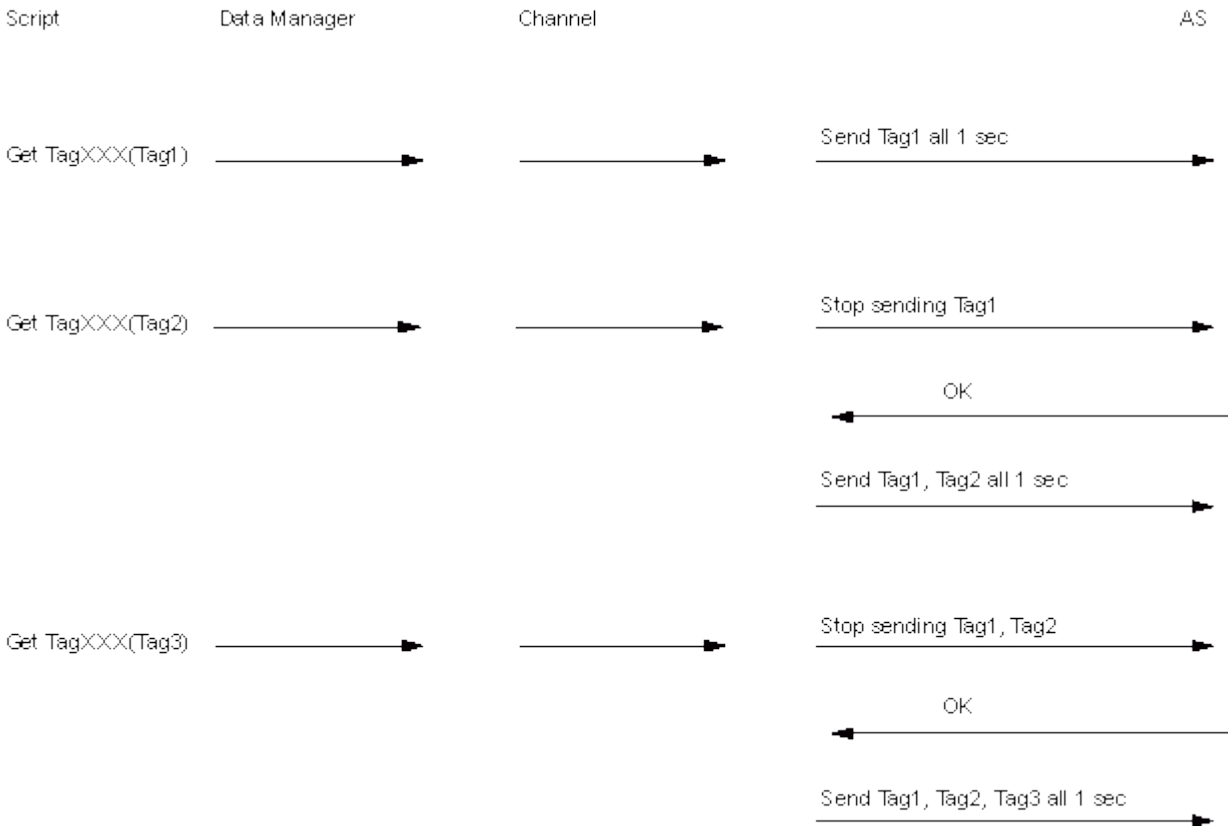
Behavior in actions with default trigger:

tags 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.

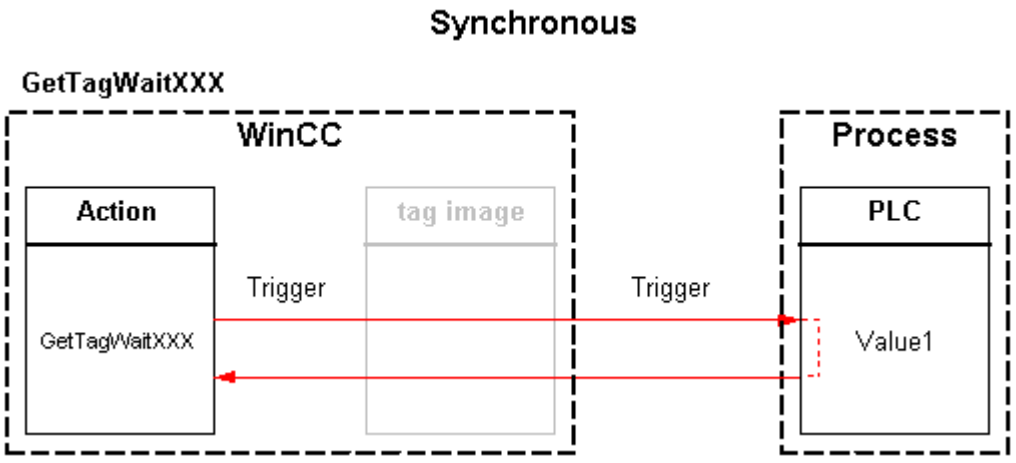
To avoid this increase in basic loading, the value can be requested by GetTagXXXWait(). The call GetTagXXXWait() causes a higher communication load one time but the tag is not added to the tag image.

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.



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.

GetTagXXXStateQCWait

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 codese of the tags.

state

wait

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

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 1543)

GetTagBitStateWait example (Page 1495)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

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 the data type "char"

See also

Functionality of the GetTag functions (Page 1355)

Tag statuses (Page 1543)

Beispiel GetTagCharStateWait (Page 1497)

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.

Return value

Value of the tag in the data type "double"

See also

Tag statuses (Page 1543)

GetTagFloatStateWait example (Page 1498)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagFloatStateWait example (Page 1498)

Functionality of the GetTag functions (Page 1355)

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)

Parameters

pdwState

Field in which the tag statuses are stored.

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 1538)

Tag statuses (Page 1543)

GetTagMultiStateWait example (Page 1500)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagRawStateWait example (Page 1503)

Functionality of the GetTag functions (Page 1355)

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"

See also

Tag statuses (Page 1543)

GetTagSByteStateWait example (Page 1505)

Functionality of the GetTag functions (Page 1355)

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

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 1543)

GetTagSByteStateWait example (Page 1505)

Functionality of the GetTag functions (Page 1355)

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 1355)

Tag statuses (Page 1543)

GetTagSByteStateWait example (Page 1505)

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 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagBitStateWait example (Page 1495)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

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 the data type "char"

See also

Tag statuses (Page 1543)

Beispiel GetTagCharStateWait (Page 1497)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagFloatStateWait example (Page 1498)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

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 1498)

Tag statuses (Page 1543)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagRawStateWait example (Page 1503)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateWait example (Page 1505)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateWait example (Page 1505)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateWait example (Page 1505)

Functionality of the GetTag functions (Page 1355)

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"

See also

Tag statuses (Page 1543)

GetTagWordStateWait example (Page 1507)

Functionality of the GetTag functions (Page 1355)

stateqc

wait

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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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".

See also

Tag statuses (Page 1543)

GetTagCharStateQCWait example (Page 1496)

Functionality of the GetTag functions (Page 1355)

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

double GetTagDoubleStateQCWait(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

Tag statuses (Page 1543)

GetTagFloatStateQCWait example (Page 1498)

Functionality of the GetTag functions (Page 1355)

GetTagDWordStateQCWait

Function

Determines 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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagFloatStateQCWait example (Page 1498)

Functionality of the GetTag functions (Page 1355)

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 1538)

Tag statuses (Page 1543)

GetTagMultiStateQCWait example (Page 1499)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagRawStateQCWait example (Page 1503)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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.

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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 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 1546)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1549)

Structure definition DM_VARKEY (Page 1550)

Functionality of the GetTag functions (Page 1355)

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".

See also

Tag statuses (Page 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagBitStateQC example (Page 1494)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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

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".

See also

Tag statuses (Page 1543)

GetTagCharStateQCWait example (Page 1496)

Functionality of the GetTag functions (Page 1355)

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

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 1498)

Tag statuses (Page 1543)

Functionality of the GetTag functions (Page 1355)

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.

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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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 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

GetTagFloatStateQCWait example (Page 1498)

Tag statuses (Page 1543)

Functionality of the GetTag functions (Page 1355)

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.

See also

Tag statuses (Page 1543)

GetTagRawStateQCWait example (Page 1503)

Functionality of the GetTag functions (Page 1355)

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

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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

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 1543)

GetTagSByteStateQCWait example (Page 1504)

Functionality of the GetTag functions (Page 1355)

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

```
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 1546)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1549)

Structure definition DM_VARKEY (Page 1550)

Functionality of the GetTag functions (Page 1355)

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 1543)

GetTagWordStateQCWait example (Page 1506)

Functionality of the GetTag functions (Page 1355)

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

GetTagBit example (Page 1493)

Functionality of the GetTag functions (Page 1355)

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 1505)

Functionality of the GetTag functions (Page 1355)

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.

See also

GetTagChar example (Page 1495)
Functionality of the GetTag functions (Page 1355)

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

```
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 1497)
Functionality of the GetTag functions (Page 1355)

GetTagDWordWait

Function

Determines the value of a tag of data type "unsigned 32 bit". The value is read explicitly from the AS.

Syntax

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 1355)

GetTagWord example (Page 1505)

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

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 1497)

Functionality of the GetTag functions (Page 1355)

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 1538)

GetTagMultiWait example (Page 1501)

Functionality of the GetTag functions (Page 1355)

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 1355)

GetTagRaw example (Page 1502)

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 1504)

Functionality of the GetTag functions (Page 1355)

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 1504)

Functionality of the GetTag functions (Page 1355)

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 1504)

Functionality of the GetTag functions (Page 1355)

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 1546)

Structure definition DM_VAR_UPDATE_STRUCT (Page 1548)

Structure definition DM_VARKEY (Page 1550)

Functionality of the GetTag functions (Page 1355)

GetTagWordWait

Function

Determines the value of a tag of data type "unsigned 16 bit". The value is read explicitly from the AS.

Syntax

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 1505)

Functionality of the GetTag functions (Page 1355)

GetTagBit

Function

Determines the value of a tag of data type "Binary tag".

Syntax

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 1493)

Functionality of the GetTag functions (Page 1355)

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 1505)

Functionality of the GetTag functions (Page 1355)

GetTagChar

Function

Determines the value of a tag of data type "8-bit text tag" or "16-bit text tag".

Syntax

char* GetTagChar(Tag Tag_Name);

Parameters

Tag_Name

name of the tag

Return value

Pointer to a character string containing the value of the tag.

See also

GetTagChar example (Page 1495)

Functionality of the GetTag functions (Page 1355)

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 1497)

Functionality of the GetTag functions (Page 1355)

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 1505)

Functionality of the GetTag functions (Page 1355)

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 1497)

Functionality of the GetTag functions (Page 1355)

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

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

GetTagRaw example (Page 1502)

Functionality of the GetTag functions (Page 1355)

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 1504)

Functionality of the GetTag functions (Page 1355)

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 1504)

Functionality of the GetTag functions (Page 1355)

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 1504)

Functionality of the GetTag functions (Page 1355)

GetTagValue

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 1546)
Structure definition DM_VAR_UPDATE_STRUCT (Page 1548)
Structure definition DM_VARKEY (Page 1550)
Functionality of the GetTag functions (Page 1355)

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 1505)

Functionality of the GetTag functions (Page 1355)

3.3.5.3 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 functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

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 1543)

Beispiel SetTagBitStateWait (Page 1522)

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.

Principle of the SetTag functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

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 1543)

Beispiel SetTagWordStateWait (Page 1529)

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 functionsEXAMPLE_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 1543)

SetTagCharStateWait example (Page 1523)

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"

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 1543)

SetTagFloatStateWait example (Page 1524)

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 functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

BOOL SetTagDWordStateWait(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. To do this, the tag status must be evaluated.

FALSE

An error has occurred.

See also

Tag statuses (Page 1543)

Beispiel SetTagWordStateWait (Page 1529)

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 functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

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 1543)

SetTagFloatStateWait example (Page 1524)

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.

Principle of the SetTag functionsEXAMPLE_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

Syntax

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.

FormatdescriberEXAMPLES_INTERNAL_FUNCTIONS_TAG_STATEWAIT_23_130

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 1543)

SetTagMultiStateWait example (Page 1524)

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 functionsEXAMPLE_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 1543)

SetTagRawStateWait example (Page 1527)

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 functionsEXAMPLE_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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

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 functionsEXAMPLE_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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

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 functionsEXAMPLE_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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

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 functionsEXAMPLE_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 1543)

Beispiel SetTagWordStateWait (Page 1529)

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 1543)

Beispiel SetTagBitStateWait (Page 1522)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagWordStateWait (Page 1529)

Principle of the SetTag functions (Page 1417)

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 1543)

SetTagCharStateWait example (Page 1523)

Principle of the SetTag functions (Page 1417)

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 1543)

SetTagFloatStateWait example (Page 1524)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagWordStateWait (Page 1529)

Principle of the SetTag functions (Page 1417)

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 1543)

SetTagFloatStateWait example (Page 1524)

Principle of the SetTag functions (Page 1417)

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.

FALSE

An error has occurred.

See also

Tag statuses (Page 1543)

GetTagRaw example (Page 1502)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagSByteStateWait (Page 1528)

Principle of the SetTag functions (Page 1417)

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 1543)

Beispiel SetTagWordStateWait (Page 1529)

Principle of the SetTag functions (Page 1417)

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 1522)

Principle of the SetTag functions (Page 1417)

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 1528)

Principle of the SetTag functions (Page 1417)

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 1417)

SetTagChar example (Page 1523)

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

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 1524)

Principle of the SetTag functions (Page 1417)

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 1528)

Principle of the SetTag functions (Page 1417)

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 1524)

Principle of the SetTag functions (Page 1417)

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 1538)

SetTagMultiWait example (Page 1525)

Principle of the SetTag functions (Page 1417)

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 1526)

Principle of the SetTag functions (Page 1417)

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 1527)

Principle of the SetTag functions (Page 1417)

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 1417)

SetTagSByte example (Page 1527)

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

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 1527)

Principle of the SetTag functions (Page 1417)

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

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 1546)

Tag statuses (Page 1543)

Structure definition DM_VAR_UPDATE_STRUCT (Page 1548)

Structure definition DM_VARKEY (Page 1550)

Principle of the SetTag functions (Page 1417)

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 1528)

Principle of the SetTag functions (Page 1417)

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 1417)

SetTagBit example (Page 1522)

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 1528)

Principle of the SetTag functions (Page 1417)

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 1523)

Principle of the SetTag functions (Page 1417)

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 1524)

Principle of the SetTag functions (Page 1417)

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 1528)

Principle of the SetTag functions (Page 1417)

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 1524)

Principle of the SetTag functions (Page 1417)

SetTagRaw

Function

Sets the value of a tag of the data type "Raw data type".

Syntax

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 1526)

Principle of the SetTag functions (Page 1417)

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 1527)

Principle of the SetTag functions (Page 1417)

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 1527)

Principle of the SetTag functions (Page 1417)

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 1527)
Principle of the SetTag functions (Page 1417)

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 1417)
Structure definition CMN_ERROR (Page 1546)
Tag statuses (Page 1543)
Structure definition DM_VAR_UPDATE_STRUCT (Page 1548)
Structure definition DM_VARKEY (Page 1550)

SetTagWord

Function

Sets the value of a tag of the data type "unsigned 16 Bit".

Syntax

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.

See also

SetTagWord example (Page 1528)
Principle of the SetTag functions (Page 1417)

3.3.6 WinCC

3.3.6.1 WinCC - short description

The functions of the WinCC group allow to define various setting in Runtime.

The functions of the System subgroup can be used to influence WinCC Runtime.

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.

3.3.6.2 system

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 1473)

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 1474)

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 1481)

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 1509)

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 1542)

SetLanguage example (Page 1517)

3.3.6.3 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.

3.3.6.4 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

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 1491)

GetServerTagPrefix example (Page 1492)

3.3.6.5 TraceText

Function

The value defined in <Parameter> is recorded in APDiag if the specified diagnostic level has been reached.

Syntax

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.

3.3.6.6 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

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.

3.4 Appendix

3.4.1 Examples

3.4.1.1 Examles - 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.

AXC_OnBtnMsgFirst example

```
{
// jump to the first message in the WinCC Alarm Control
AXC_OnBtnMsgFirst("gs_alarm_00","Control1");
}
```

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.

DeactivateRTProject example

```
{
//deactivate the runtime
DeactivateRTProject ();
}
```

This function deactivates WinCC Runtime.

ExitWinCC example

```
{
//exit wincc
ExitWinCC ();
}
```

This function exits WinCC.

3.4.1.2 Examples - GetAlarmHigh to GetPropChar

GetAlarmHigh example

```
{
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

```
{
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 temporary store in bfilling.
2. Executing user-defined code, depending on the return value of the function.

GetFillingIndex example

```
{
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

```
{
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

```
{
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

```
{
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 obvject and and temporarily store it in flashcol_on.
2. Executing user-defined code for processing return values.

GetFlashRateFlashPic example

```
{
long lFlashRate;

//Get the flashrate
lFlashRate = GetFlashRateFlashPic(lpszPictureName,"StatusDisplay1");

//User defined code where the
//user can do something with the return value
...
}
```

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

```
{
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

```
{
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

```
{
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

```
{
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

```
{
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\r\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

```
{
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

```
{
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

```
{
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

```
{
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

```
{
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.

3.4.1.3 Examples - GetRangeMax to GetWidth

GetRangeMax example

```
{
long int lrange;
```

```
//Get the upper scale Limit
lrange = GetRangeMax(lpszPictureName,"Control1");

//User defined code where the
//user can do something with the return value
...
}
```

Parameters of the GetRangeMax 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 upper limit of the object and temporarily store it in lrange.
2. Executing user-defined code for processing return values.

GetRangeMin example

```
{
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:

"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 lower limit of the object and temporarily store it in lrange.
2. Executing user-defined code for processing return values.

Beispiel GetScaling

```
{
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

```
{
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 lenght 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 ist 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

```
{
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 = GetTagBitStateQCWait("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

```
{
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

```
{
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

```
{
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

```
{
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

```
{
#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

```
{
#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

```
{
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

```
{
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 GetTagSByteStateQCWait 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

```
{
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

```
{
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 then 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

```
{
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

```
{
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 lPos.
2. Executing user-defined code for processing return values.

GetVisible example

```
{
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.

3.4.1.4 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

```
{
//start the program calc.exe
ProgramExecute("C:\\Winnt\\system32\\calc.exe");
}
```

As parameter the file is to be specified with its path.

ResetFilter example

```
{
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.

RPTJobPreview example

```
{
//Start the print preview of the specified print job
RPTJobPreview("Documentation Text Library");
}
```

Parameters of the RPTJobPreview function:

@Text library (compact) is the name of the print job.

RPTJobPrint example

```
{
//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

```
char* main(...);
{
char* returnwert;
char text[17];
returnwert=SysMalloc(17);
strcpy(returnwert,&text[0];
return returnwert;
}
```

3.4.1.5 Examples - SetAlarmHigh to SetPropChar

SetAlarmHigh example

```
{
//Set the upper limit for the warning
SetAlarmHigh(lpszPictureName,"Bar1",3.0);
}
```

Parameters of the SetAlarmHigh funtion:

"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

```
{
//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

```
{
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

```
{
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

```
{
//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

```
{
//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

```
{
//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

```
{
//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

```
{
//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.

SetFlashBackColorOn 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

```
{
//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

```
{
//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

```
{
//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

```
{
//German
SetLanguage(0x0407);
}
```

The Runtime language is set to German.

SetLeft example

```
{
//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

```
{
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

```
{
//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

```
{
//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

```
{
//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

```
{
//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

```
{
//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.

3.4.1.6 Examples - SetRangeMax to SetWidth

SetRangeMax example

```
{
//Set the Upper Scale Limit
SetRangeMax(lpszPictureName,"Control1",80);
}
```

Parameters of the SetRangeMax function:

"lpszPictureName" is the name of the picture in which the object was configured.

"Control1" is the name of the object.

"80" is the upper limit to be assigned to the object.

SetRangeMin example

```
{
//Set the lower Scale Limit
SetRangeMin(lpszPictureName,"Control1",0);
}
```

Parameters of the SetRangeMin function:

"lpszPictureName" is the name of the picture in which the object was configured.

"Control1" is the name of the object.

"0" is the lower limit to be assigned to the object.

SetScaling example

```
{
//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.l
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

```
{
#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

```
{
//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

```
{
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:

"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 funtion:

"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

```
{
//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

```
{
//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

```
{
//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.

3.4.1.7 Examples - T to Z

TlgGetNumberOfColumns example

```
{
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.

TlgGetNumberOfColumns example

```
{
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

```
{
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 systemtime
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

```
{
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

```
{
//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.

3.4.2 Lists

3.4.2.1 Bar direction

Bar direction	Numeric value
Up	0
bottom	1
left	2
right	3

3.4.2.2 Bar Scaling

Numeric value	Bar Scaling
0	Linear (same weighting)
1	Logarithmic (low values emphasized)
2	Negative logarithmic (high values emphasized)
3	Automatic (linear)

Numeric value	Bar Scaling
4	Tangential (high and low values emphasized)
5	Square (high values emphasized)
6	Cubic (high values strongly emphasized)

3.4.2.3 Flash frequencies

Numeric value	Flash frequency
0	0.5 Hz
1	2 Hz
2	8 Hz

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.

3.4.2.4 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:

Characters	Meaning	Note
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:

Number	Format	Representation
123,455	999,999	123,455
123,455	999,99	123,46
123,455	9999,9999	123,4550
123,455	s09999.9999	+0123,4550
123,455	9.99999e	1.23455e+002

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.

3.4.2.5 I/O field, data type of the field content

Data type	Numeric value
Binary	0
decimal	1
string	2
hexadecimal	3

3.4.2.6 I/O field, field type

Type	Numeric value
Edition	0
Input	1
Output and input	2

3.4.2.7 Element alignment in check boxes and radio boxes

Alignment	Numeric value
left	0
right	-1

3.4.2.8 Color chart

The 16 primary colors are:

Color	Color value (Hex)	symbolic constant
Red	0x000000FF	CO_RED
Dark red	0x00000080	CO_DKRED
Green	0x0000FF00	CO_GREEN
Dark green	0x00008000	CO_DKGREEN
Blue	0x00FF0000	CO_BLUE
Dark blue	0x00800000	CO_DKBLUE
Cyan	0x00FFFF00	CO_CYAN
Dark cyan	0x00808000	CO_DKCYAN
Yellow	0x0000FFFF	CO_YELLOW
Dark yellow	0x00008080	CO_DKYELLOW
Magenta	0x00FF00FF	CO_MAGENTA
Dark magenta	0x00800080	CO_DKMAGENTA
Light gray	0x00C0C0C0	CO_LTGRAY
Gray	0x00808080	CO_DKGRAY
Black	0x00000000	CO_BLACK
White	0x00FFFFFF	CO_WHITE

Note

The symbolic constants are externally predefined by #define and provied by WinCC.

3.4.2.9 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:

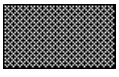
Variable	Format	C-tag
Bit	%d	DWORD / long int signed
Byte	%d	DWORD / long int signed
SByte	%d	DWORD / long int signed
Word	%d	DWORD / long int signed
SWord	%d	DWORD / long int signed
DWord	%d	DWORD / long int signed
SDWord	%d	DWORD / long int signed
Float	%f	double
Double	%f	double
Char	%s	char*

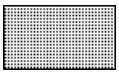
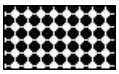
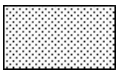
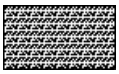
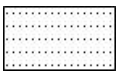
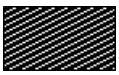
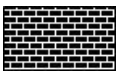
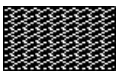
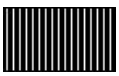
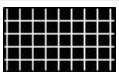
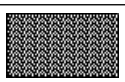
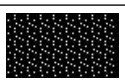
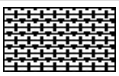
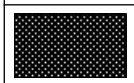
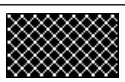
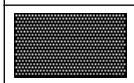
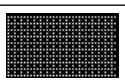
Note

If a "DWORD," for which the 32nd bit is set, is to be read, a format descriptor must be used for unsigned integers (%u).

3.4.2.10 Fill pattern

Fill pattern	Value
Transparent	65536
Solid	0

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
	1048576		196611		196627
	1048577		196612		196628

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
	1048578		196613		196629
	1048579		196614		196630
	1048832		196615		196631
	1048833		196616		196632
	1048834		196617		196633
	1048835		196618		196634
	131072		196619		196635
	131073		196620		196636
	131074		196621		196637
	131075		196622		196638
	131076		196623		196639
	196608		196624		196640
	196609		196625		196641
	196610		196626		196642

Note

The "Solid" fill pattern fills the object with the set background color.

3.4.2.11 Line styles

Line style	symbolic name	Value
	LS_SOLID	0
	LS_DASH	1
	LS_DOT	2
	LS_DASHDOT	3
	LS_DASHDOT DOT	4
hidden	LS_INVISIBLE	5

3.4.2.12 Line end style

Line end	symbolic name	Value for the left line ends	Value for the right line ends
	LE_NO	0	0
	LE_HOLLOW_ARROW	1	65536
	LE_FULL_ARROW	2	131072
	LE_CFULL_ARROW	3	196608
	LE_LINE	4	262144
	LE_HOLLOW_CIRCLE	5	327680
	LE_FULL_CIRCLE	6	393216

Note

From a line width > 5 the line end "empty circle" is displayed as filled circle.

3.4.2.13 List types

List type	Numeric value
decimal	0
Binary	1
bit	2

3.4.2.14 Language ID

WinCC only supports the SUBLANG_DEFAULT languages of Windows.

symbolic name	Value (hexadecimal)	Abbreviation
LANG_ARABIC	0x0401	
LANG_AFRIKAANS	0x0436	
LANG_ALBANIAN	0x041C	
LANG_BASQUE	0x042D	
LANG_BULGARIAN	0x0402	
LANG_BYELORUSSIAN	0x0423	
LANG_CATALAN	0x0403	
LANG_CHINESE	0x0404	
LANG_CROATIAN	0x041A	
LANG_CZECH	0x0405	CSY
LANG_DANISH	0x0406	DAN
LANG_DUTCH	0x0413	NLD
LANG_ENGLISH	0x0409	ENU
LANG_ESTONIAN	0x0425	
LANG_FAEROESE	0x0438	
LANG_FARSI	0x0429	
LANG_FINNISH	0x040B	FIN
LANG_FRENCH	0x040C	FRA
LANG_GERMAN	0x0407	DEU
LANG_GREEK	0x0408	
LANG_HEBREW	0x040D	
LANG_HUNGARIAN	0x040E	HUN
LANG_ICELANDIC	0x040F	ISL
LANG_INDONESIAN	0x0421	
LANG_ITALIAN	0x0410	ITA
LANG_JAPANESE	0x0411	
LANG_KOREAN	0x0412	
LANG_LATVIAN	0x0426	
LANG_LITHUANIAN	0x0427	
LANG_NORWEGIAN	0x0414	NOR
LANG_POLISH	0x0415	PLK
LANG_PORTUGUESE	0x0416	PTB
LANG_ROMANIAN	0x0418	
LANG_RUSSIAN	0x0419	RUS
LANG_SLOVAK	0x041B	SKY
LANG_SLOVENIAN	0x0424	
LANG_SORBIAN	0x042E	
LANG_SPANISH	0x040A	ESP

symbolic name	Value (hexadecimal)	Abbreviation
LANG_SWEDISH	0x041D	SVE
LANG_THAI	0x041E	
LANG_TURKISH	0x041F	TRK
LANG_UKRAINIAN	0x0422	

3.4.2.15 Text alignment

Horizontal	Alignment	Numeric value
	left	0
	centered	1
	right	2

Vertical	Alignment	Numeric value
	Up	0
	centered	1
	bottom	2

3.4.2.16 Tag statuses

Value (decimal)	Value (hexdecimal)	Meaning
0	0x0000	No error
1	0x0001	Connection to partner not established
2	0x0002	Handshake error
4	0x0004	Network module defective
8	0x0008	Configured upper limit exceeded
16	0x0010	Configured lower limit exceeded
32	0x0020	Format upper limit exceeded
64	0x0040	Format lower limit exceeded
128	0x0080	Conversion error
256	0x0100	Tag initialization value
512	0x0200	Tag replacement value
1024	0x0400	Channel addressing error
2048	0x0800	Tag not found or not available

Value (decimal)	Value (hexdecimal)	Meaning
4096	0x1000	Access to tag not permitted
8192	0x2000	Timeout, no check-back message from the channel
16384	0x4000	Server not available.

3.4.3 Structure definitions

3.4.3.1 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:

dwErrorCode1	dwErrorCode2	bCycle	szApplicationName	szFunctionName	szTagName	lpParamSize	dwParamSize	dwCycle	pError	Description
1007001	0	X	X	X		X	X			Action requires exception
1007001	1	X	X	X		X	X			Exception when accessing the return result
1007001	4097	X	X	X		X	X			Stack overflow while executing the action

1007001	4098	X	X	X		X	X			The action contains a division by 0
1007001	4099	X	X	X		X	X			The action contains an access to a non-existing symbol
1007001	4100	X	X	X		X	X			The action contains an access violation
1007004	0	X	X	X						Function is not known
1007005	1	X	X							Action does not include a P code.
1007005	2	X	X							Incorrect function name
1007005	4	X	X	X		X	X			Return value type is invalid
1007005	32768ff	X	X	X		X	X			Ciss Compiler error when loading the action
1007006	0	X	X	X	X	X	X	X		Tag is not defined
1007006	1	X	X	X	X	X	X	X		Tag timeout
1007006	2	X	X	X	X	X	X	X	X	Tag cannot be returned in the desired format
1007006	3	X	X	X	X	X	X	X	X	Tag returns status violation, status present in CMN_ERROR.dwError1
1007007	1	X	X	X		X	X		X	Error in PDLRTGetProp
1007007	2	X	X	X		X	X		X	Error in PDLRTSetProp
1007007	3	X	X	X		X	X		X	Error with DM call

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 1546)

3.4.3.2 Structure definition CCAPTime

```
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:

dwCode = 113	Call with time definition for each action
dwCode = 114	Call with time monitoring for each action

dwFlags

The structure element dwFlags provides information on the output type:

dwFlags = TRUE	The results are output to a file
dwFlags = FALSE	The results are output to the diagnostics window

3.4.3.3 Structure definition CMN_ERROR

```
struct CMNERRORSTRUCT {
DWORD       dwError1,
DWORD       dwError2,
DWORD       dwError3,
DWORD       dwError4,
DWORD       dwError5;
TCHAR       szErrorText[MAX_ERROR_LEN];
}
CMN_ERROR
```

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.

szErrorText

Buffer for the text description of the error cause

The content is determined from the resources and therefore language-dependent.

3.4.3.4 Structure definition DM_TYPEREF

```
typedef struct {
DWORD dwType;
DWORD dwSize;
char szTypeName[MAX_DM_TYPE_NAME + 1];
}
DM_TYPEREF;
```

Members

dwType

Specifies the tag type

DM_VARTYPE_BIT	Binary tag
DM_VARTYPE_SBYTE	Signed 8-bit value
DM_VARTYPE_BYTE	Unsigned 8-bit value
DM_VARTYPE_SWORD	Signed 16-bit value
DM_VARTYPE_WORD	Unsigned 16-bit value
DM_VARTYPE_SDWORD	Signed 32-bit value
DM_VARTYPE_DWORD	Unsigned 32-bit value
DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754

DM_VARTYPE_TEXT_8	Text tag, 8-bit font
DM_VARTYPE_TEXT_16	Text tag, 16-bit font
DM_VARTYPE_RAW	Raw data type
DM_VARTYPE_STRUCT	Structure tag
DM_VARTYPE_TEXTREF	Text reference tag

dwSize

Specifies the length of the data type in bytes.

szTypeName

In the case of structure tags, contains the name of the structure type

3.4.3.5 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 1543)

Structure definition DM_VARKEY (Page 1550)

Structure definition DM_TYPEREF (Page 1547)

3.4.3.6 Structure definition DM_VAR_UPDATE_STRUCTEX

```
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.

```
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 1543)

Structure definition DM_VARKEY (Page 1550)

Structure definition DM_TYPEREF (Page 1547)

3.4.3.7 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

3.4.3.8 Structure definition LINKINFO

```
typedef struct {
LINKTYPE LinkType;
DWORD dwCycle;
TCHAR szLinkName[256];
}
LINKINFO;
```

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.

BUBRT_LT_NOLINK	0	no shortcut
BUBRT_LT_VARIABLE_DIRECT	1	direct tag
BUBRT_LT_VARIABLE_INDIRECT	2	indirect tag
BUBRT_LT_ACTION	3	C action
BUBRT_LT_ACTION_WIZARD	4	Dynamic Dialog
BUB_LT_DIRECT_CONNECTION	5	Direct connection
BUBRT_LT_ACTION_WIZARD_INPROC	6	Dynamic Dialog

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

255	Picture cycle
235	Window Cycle
0	Upon change
1	250ms
2	500 ms
3	1 s
4	2 s
5	5s
6	10s
7	1min
8	5min
9	10min
10	1h
11-15	User cycle 1-5

szLinkName

Tag name

3.4.3.9 Structure definition MSG_FILTER_STRUCT

```
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       dwArchivMode;
char        szTB[MSG_MAX_TB][
MSG_MAX_TB_CONTENT+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":

MSG_FILTER_DATE_FROM	Date from
MSG_FILTER_DATE_TO	Date to
MSG_FILTER_TIME_FROM	Time from
MSG_FILTER_TIME_TO	Time to
MSG_FILTER_NR_FROM	Message number from
MSG_FILTER_NR_TO	Message number to
MSG_FILTER_CLASS	Message classes
MSG_FILTER_STATE	Message status
MSG_FILTER_AG_FROM	AS number from
MSG_FILTER_AG_TO	AS number to
MSG_FILTER_AGSUB_FROM	AG subnumber from
MSG_FILTER_AGSUB_TO	AG subnumber to
MSG_FILTER_TEXT	Message texts
MSG_FILTER_PVALUE	Process values

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:

MSG_FILTER_QUICK_MONTH	Quick selection last n months
MSG_FILTER_QUICK_DAYS	Quick selection last n days
MSG_FILTER_QUICK_HOUR	Quick selection last n hours

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 calucalted back n * (months, days, hours).

3.4.3.10 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

MSG_STATE_COME	0x00000001	Message came in
MSG_STATE_GO	0x00000002	Message went out
MSG_STATE_QUIT	0x00000003	Message acknowledged
MSG_STATE_LOCK	0x00000004	Message locked
MSG_STATE_UNLOCK	0x00000005	Message unlocked
MSG_STATE_QUIT_SYSTEM	0x00000010	Message acknowledged by system
MSG_STATE_QUIT_EMERGENCY	0x00000011	Emergency acknowledgement
MSG_STATE_QUIT_HORN	0x00000012	Horn acknowledgement
MSG_STATE_COMEGO	0x00000013	Message came in and went out, only display in message list
MSG_STATE_UPDATE	0x00010000	Bit for message update
MSG_STATE_RESET	0x00020000	Bit for message reset
MSG_STATE_SUMTIME	0x00040000	Bit active for daylight savings time
MSG_STATE_INSTANCE	0x00080000	Bit for instance message (n messages of a no.)

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

MSG_FLAG_SUMTIME	0x00000001	Daylight savings time active
MSG_FLAG_COMMENT	0x00000002	Message has comments
MSG_FLAG_ARCHIV	0x00000004	Archiving
MSG_FLAG_PROTOCOL	0x00000008	Logging
MSG_FLAG_TEXTVALUES	0x00000010	Message has values accompanying the text
MSG_FLAG_TIMEINVALID	0x00000020	Bit for invalid date/time stamp
MSG_FLAG_INSTANCE	0x00000040	Instance message identification (185269)

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.

VBA for Automated Configuration 4

4 Resources

4.1 Automated Configuring

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.

4.2 Introduction: Using VBA in WinCC

4.2.1 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 1562)

VBA Reference (Page 1664)

VBA in the Graphics Designer (Page 1567)

Executing VBA Macros in Graphics Designer (Page 1566)

Differentiation: Using VBA (Page 1561)

4.2.2 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 1567)

Executing VBA Macros in Graphics Designer (Page 1566)

Organizing VBA Code in a WinCC Project (Page 1562)

Introduction: Using VBA in WinCC (Page 1560)

4.2.3 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.

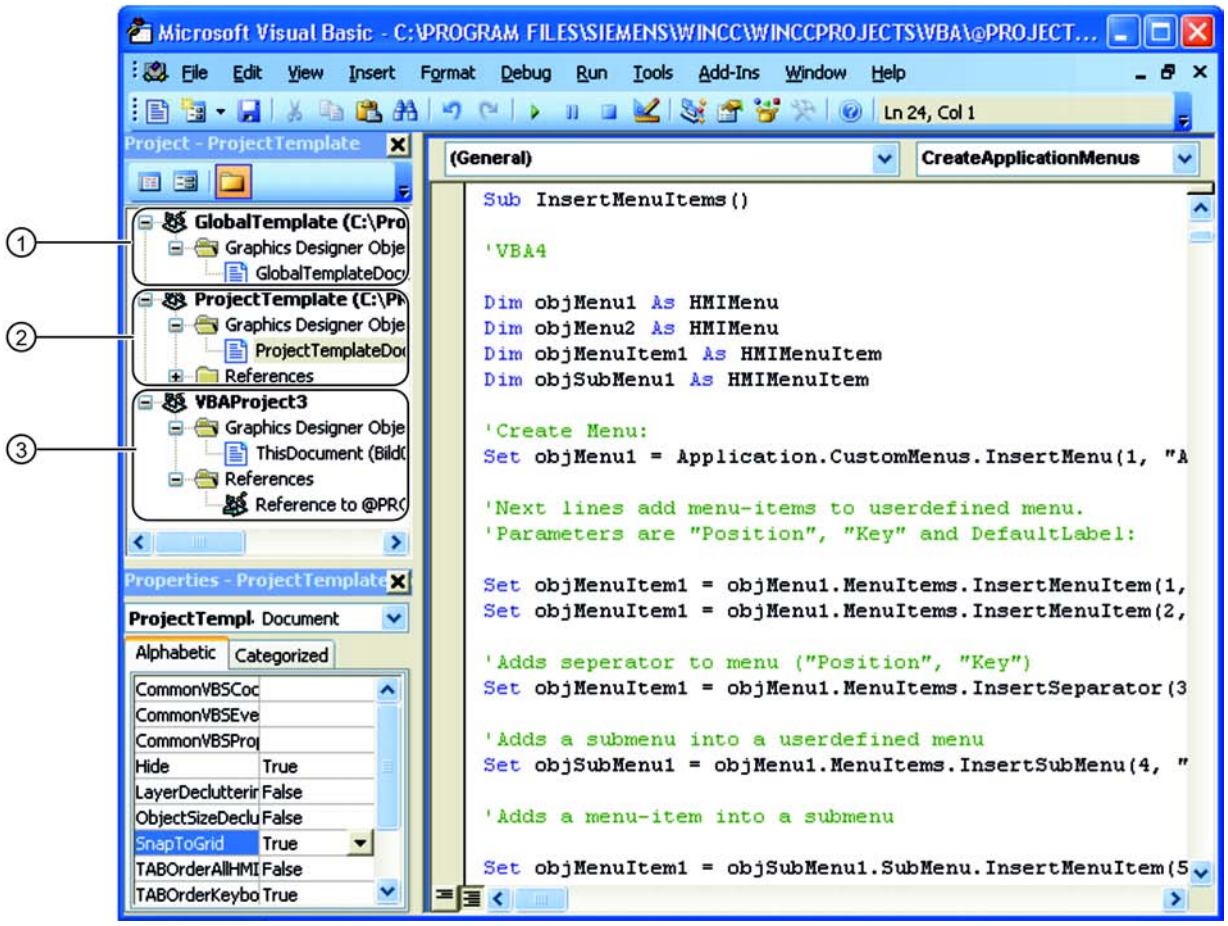
Note

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 1655)

VBA in the Graphics Designer (Page 1567)

Executing VBA Macros in Graphics Designer (Page 1566)

How to export and import VBA code (Page 1565)

4.2.4 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 1562)

4.2.5 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:

```
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 1664)

Event Handling (Page 1655)

Creating Customized Menus and Toolbars (Page 1573)

VBA in the Graphics Designer (Page 1567)

Organizing VBA Code in a WinCC Project (Page 1562)

Introduction: Using VBA in WinCC (Page 1560)

4.3 VBA in the Graphics Designer

4.3.1 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.

See also

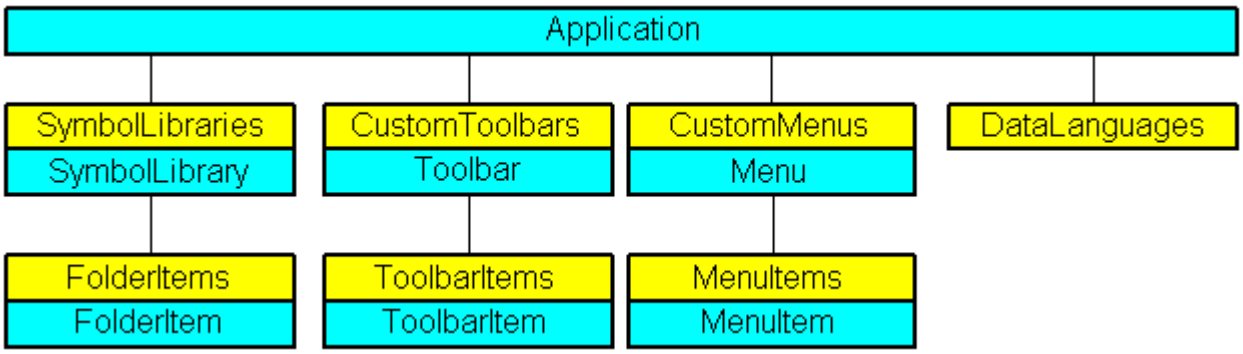
Editing Pictures with VBA (Page 1601)
SymbolLibrary Object (Page 1962)
HMIObject Object (Page 1882)
Document Object (Page 1847)
Application Object (Page 1814)
Accessing External Applications with VBA (Page 1658)
Event Handling (Page 1655)
Creating Dynamics with VBA (Page 1631)
Editing Objects with VBA (Page 1606)
Adapting the Graphics Designer with VBA (Page 1569)
Introduction: Using VBA in WinCC (Page 1560)

4.3.2 Adapting the Graphics Designer with VBA

4.3.2.1 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 1601)

Accessing the component library with VBA (Page 1593)

Creating Customized Menus and Toolbars (Page 1573)

Language-Dependent Configuration with VBA (Page 1570)

VBA in the Graphics Designer (Page 1567)

4.3.2.2 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 by replacing language-dependent tool icons, for example.

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".

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 full list of language codes is available in the WinCC documentation (Index > Language ID).

In this example a German label and an English label are assigned to the button "myButton":

```
Sub AddLanguagesToButton()
'VBA2
Dim objLabelText As HMILanguageText
Dim objButton As HMIButton
```

```
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("myButton", "HMIButton")
'
'Set defaultlabel:
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 1893)

LanguageFonts Object (Listing) (Page 1890)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Create Menus in Multiple Languages (Page 1580)

VBA Reference (Page 1664)

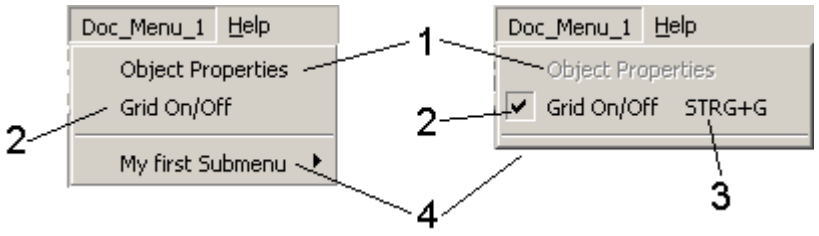
4.3.2.3 Creating Customized Menus and Toolbars

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:



Active (yes/no) (1)

Activates the entry or dims it. You can use the "Enabled" property for user-defined menus, menu items and icons:

```
'VBA13
Application.ActiveDocument.CustomMenus(1).MenuItems(1).Enabled = False
```

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:

```
'VBA14
Application.ActiveDocument.CustomMenus(1).MenuItems(2).Checked = True
```

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:

```
'VBA15
Application.ActiveDocument.CustomMenus(1).MenuItems(3).Shortcut = "Ctrl+G"
```

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
Application.ActiveDocument.CustomMenus(1).MenuItems(4).Visible = False
```

Creating Customized Menus and Toolbars

User-defined menus and toolbars in the Graphics Designer

You can use user-defined menus and toolbars in the Graphics Designer in order to execute VBA macros. A distinction is drawn here between application-specific and picture-specific menus and toolbars, which have the following properties:

- Application-specific menu/toolbar: 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 menu/toolbar: Is linked with a specific picture and remains visible as long as the picture is visible. 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. However:

- application-specific menus are always positioned to the right of the "Window" menu in the Graphics Designer, whereas

picture-specific menus are always positioned to the left of the "Help" menu in the Graphics Designer.

However, application-specific toolbars are not treated "with preference": In this case, the positioning is determined by the order in which you insert the toolbars. Toolbars are positioned beneath the Graphics Designer toolbar:

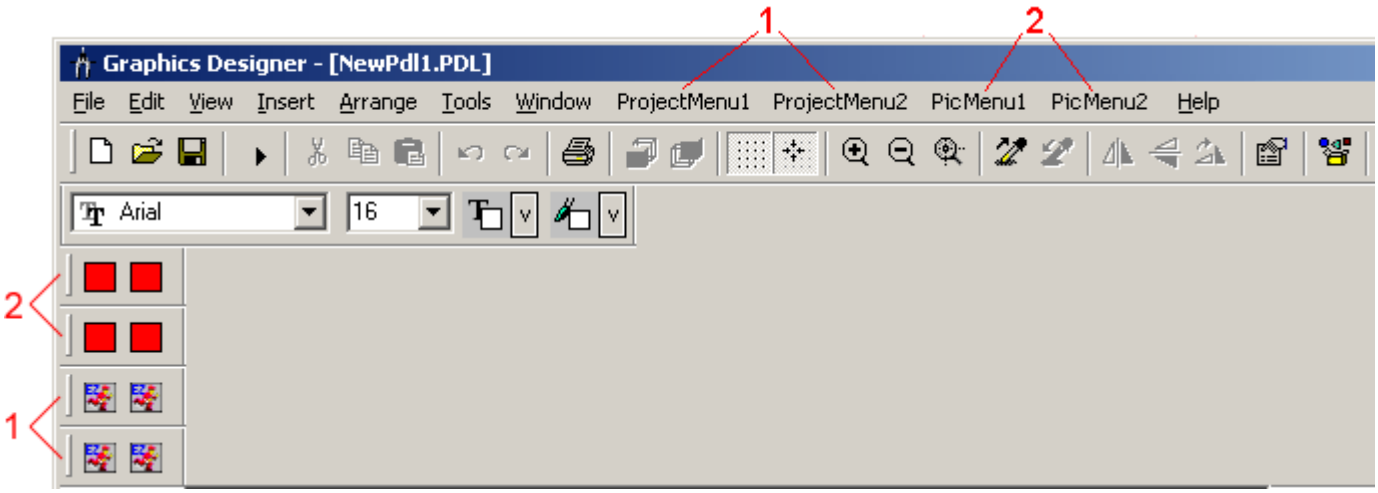


Figure 4-1 Figure: User-defined menus and toolbars. (1) Application-specific, (2) picture-specific

In the illustration above, first the picture-specific menus and toolbars are inserted, then the application-specific ones.

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 item (Checked property).
- Shortcut: Key combination for calling a menu item (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 item, 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 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Add a New Icon to the Toolbar (Page 1584)

How to Create a New Application-Specific Toolbar (Page 1582)

How to Create Menus in Multiple Languages (Page 1580)

How to Add a New Menu Item to the Menu (Page 1577)

How to Create Picture-Specific Menus and Toolbars (Page 1602)

How to Create a New Application-Specific Menu (Page 1575)

Configuring Menus and Toolbars (Page 1572)

Executing VBA Macros in Graphics Designer (Page 1566)

How to Create a New Application-Specific Menu

Introduction

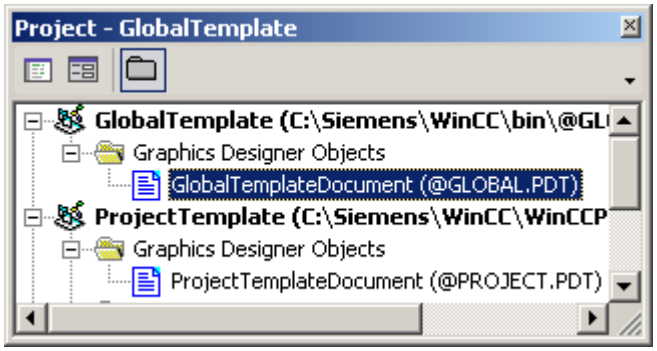
Application-specific menus remain visible even when all pictures in Graphics Designer are closed. You can use the Started event, for example, in order to insert an application-specific menu at an early stage.

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:



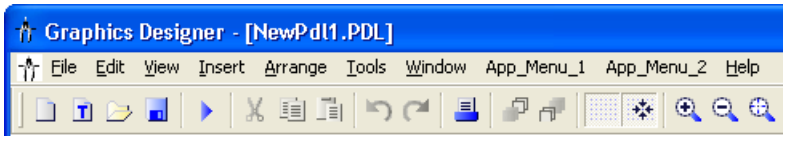
3. To create a user-defined menu in the Graphics Designer, you can for example insert a "CreateApplicationMenus()" procedure in the document. In this example, two user-defined menus are created:

```
Sub CreateApplicationMenus()
'VBA3
'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")
Set objMenu2 = Application.CustomMenus.InsertMenu(2, "AppMenu2",
"App_Menu_2")
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:



See also

Creating Customized Menus and Toolbars (Page 1573)

InsertMenu Method (Page 1763)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Create Menus in Multiple Languages (Page 1580)

How to Add a New Menu Item to the Menu (Page 1577)

Configuring Menus and Toolbars (Page 1572)

Organizing VBA Code in a WinCC Project (Page 1562)

How to Add a New Menu Item to the 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:

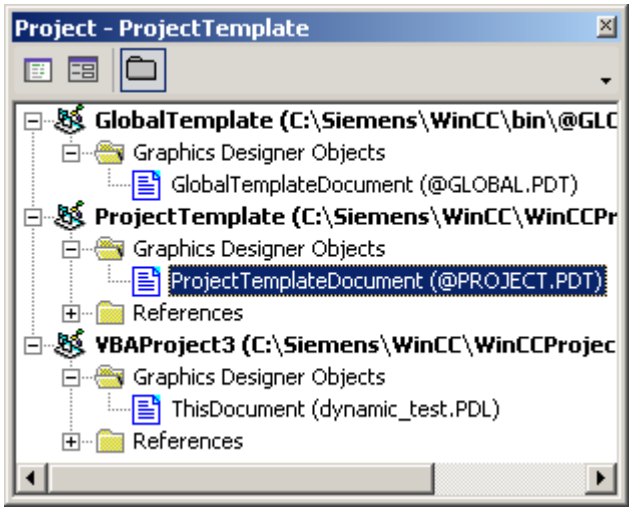
- Menu entry: To call VBA macros.
- Separator line: For clearer design of user-defined menu.
- Submenu: Same as user-defined menu (e.g. command structuring).

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:



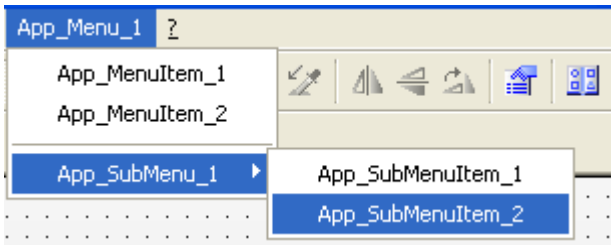
3. To create menu items in a previously created user-defined menu, you can for example insert an "InsertMenuItems()" procedure in the document. In this example a number of menu items are created in the user-defined menu "App_Menu_1":

```
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:



See also

InsertSeparator Method (Page 1766)

InsertSubmenu Method (Page 1767)

InsertMenu Method (Page 1763)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Create Menus in Multiple Languages (Page 1580)

How to Create a New Application-Specific Menu (Page 1575)

Configuring Menus and Toolbars (Page 1572)

Creating Customized Menus and Toolbars (Page 1573)

How to Create Menus in Multiple Languages

Introduction

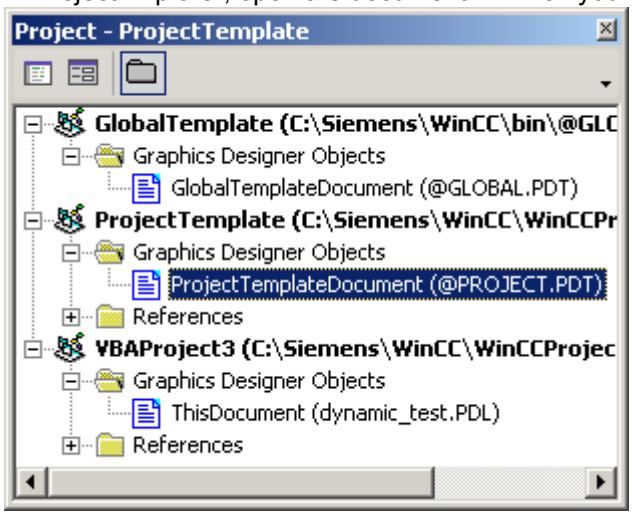
You can create a user-defined menu that responds to a change of language. To do this you need to define the necessary number of labels in other languages for the menu and for each menu item.

The foreign-language label comprises the language ID (LCID) and the foreign-language text (DisplayName).

The full list of language codes is available in the WinCC documentation (Index > Language ID).

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:

```
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 objLanguageTextMenuItem1 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 1893)

LDLabelTexts Property (Page 2186)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Add a New Menu Item to the Menu (Page 1577)

How to Create a New Application-Specific Menu (Page 1575)

Configuring Menus and Toolbars (Page 1572)

Creating Customized Menus and Toolbars (Page 1573)

Language-Dependent Configuration with VBA (Page 1570)

How to Create a New Application-Specific Toolbar

Introduction

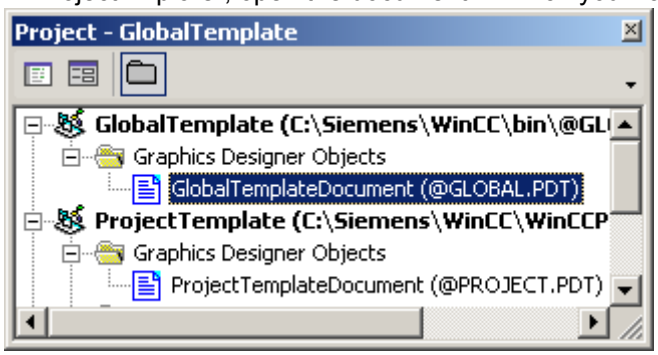
Application-specific toolbars remain visible even when all pictures in the Graphics Designer are closed.

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:

```
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 1708)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Add a New Icon to the Toolbar (Page 1584)

Configuring Menus and Toolbars (Page 1572)

Creating Customized Menus and Toolbars (Page 1573)

How to Add a New Icon to the Toolbar

Requirements

You must have created the user-defined toolbar first.

Introduction

You can insert two different types of objects in the user-defined toolbar:

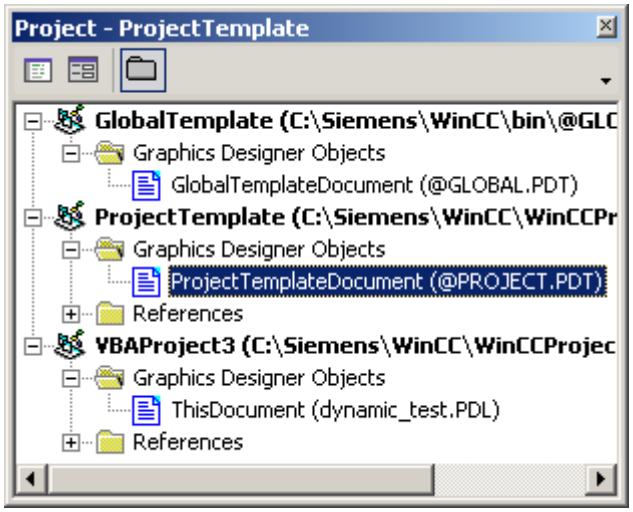
- Symbol: To call VBA macros.
- Separator line: For clearer design of user-defined toolbars.

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:



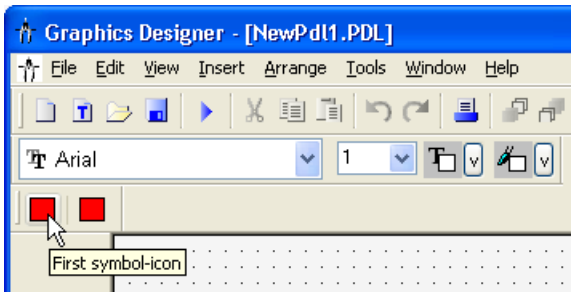
3. To create icons in a previously created user-defined toolbar, you can for example insert an "InsertToolbarItems()" procedure in the document. In this example, two icons separated by a separator line are created in the user-defined toolbar "AppToolbar1":

```
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 seperator 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>.

Results

The "InsertToolbarItems()" procedure adds a toolbar with two icons, separated by a dividing line, to the Graphics Designer toolbars:



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 1573)

Icon Property (Page 2144)

InsertSeparator Method (Page 1766)

InsertToolbarItem Method (Page 1769)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Create a New Application-Specific Toolbar (Page 1582)

Configuring Menus and Toolbars (Page 1572)

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

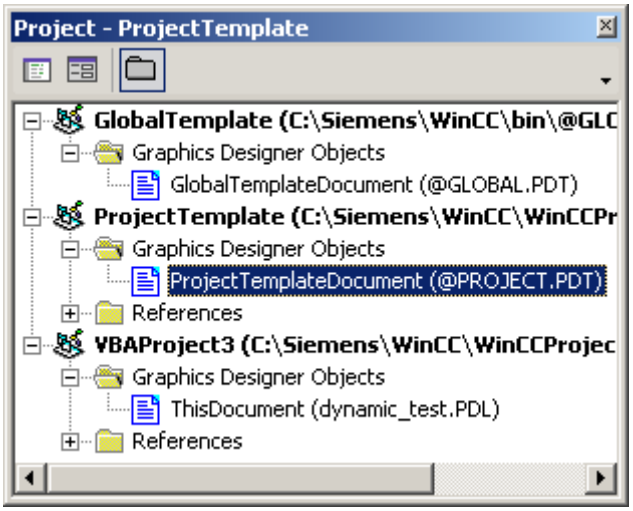
When the configuring engineer moves the mouse over a user-defined menu item or over a user-defined icon, you can provide additional help text to explain the functionality in more detail:

- 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 (<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 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 HMIMenu
'
'Object "objStatusTextMenuItem1" contains foreign-language texts
Dim objStatusTextMenuItem1 As HMILanguageText
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 objStatusTextMenuItem1 =
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:

```
Sub AddStatusAndTooltipTextsToAppToolbar1()
'VBA9
Dim objToolbar1 As HMIToolbar
'
'Variable "StatusTextToolbarItem1" for foreign statustexts
Dim objStatusTextToolbarItem1 As HMILanguageText
'
'Variable "TooltipTextToolbarItem1 for foreign tooltiptexts
Dim objTooltipTextToolbarItem1 As HMILanguageText
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 2190)

LDStatusTexts Property (Page 2188)

LanguageTexts Object (Listing) (Page 1893)

Add Method (Page 1705)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Add a New Icon to the Toolbar (Page 1584)

How to Add a New Menu Item to the Menu (Page 1577)

Configuring Menus and Toolbars (Page 1572)

Creating Customized Menus and Toolbars (Page 1573)

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 "ToolbarItemClicked" 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".

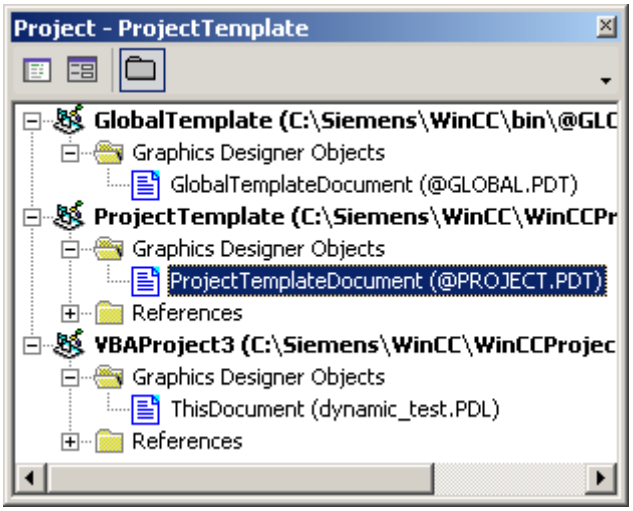
Procedure

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:

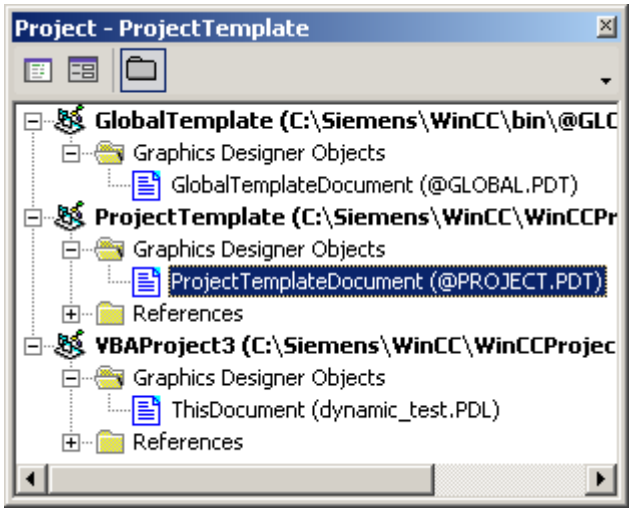


3. To start a VBA macro via the VBA event handlers, use the "MenuItemClicked" or "ToolbarItemClicked" event:

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)

```
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 2206)

ToolbarItemClicked Event (Page 1701)

MenuItemClicked Event (Page 1695)

How to Add a New Icon to the Toolbar (Page 1584)

How to Add a New Menu Item to the Menu (Page 1577)

Event Handling (Page 1655)

Executing VBA Macros in Graphics Designer (Page 1566)

4.3.2.4 Accessing the component library with VBA

Accessing the component library with VBA

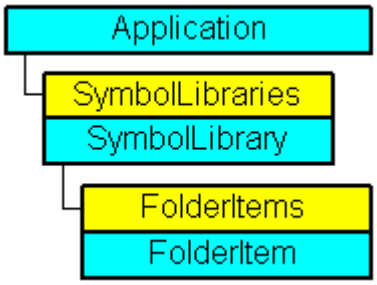
Introduction

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



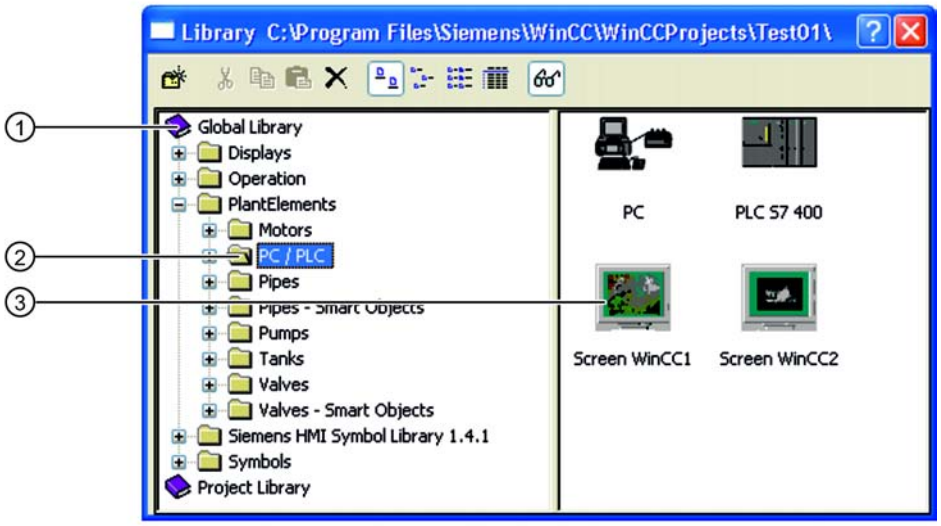
The component library is represented in VBA by the "SymbolLibraries" listing. The listing contains two elements, which represent the "Global Library" and the "Project Library". The "FolderItems" listing contains elements, which represent folders as well as objects.

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:

```
'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:

```
'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:

```
'VBA19
Application.SymbolLibraries(1).FolderItems("Folder2").Folder("Folder2").Folder.Item("Objec
t1").DisplayName
```

Creating or deleting folders in the component library

Use the following methods to create or delete folders:

- "AddFolder(DefaultName)" 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(DefaultName, 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 1740)
PasteClipboard Method (Page 1781)
GetItemByPath Method (Page 1760)
FindByDisplayName Method (Page 1756)
Delete Method (Page 1745)
AddItem Method (Page 1720)
AddFromClipboard Method (Page 1717)
AddFolder Method (Page 1716)
SymbolLibrary Object (Page 1962)
SymbolLibraries Object (Listing) (Page 1963)
How to Insert an Object from the Component Library into a Picture with VBA (Page 1599)
How to Edit the Component Library with VBA (Page 1596)

How to Edit the Component Library with VBA

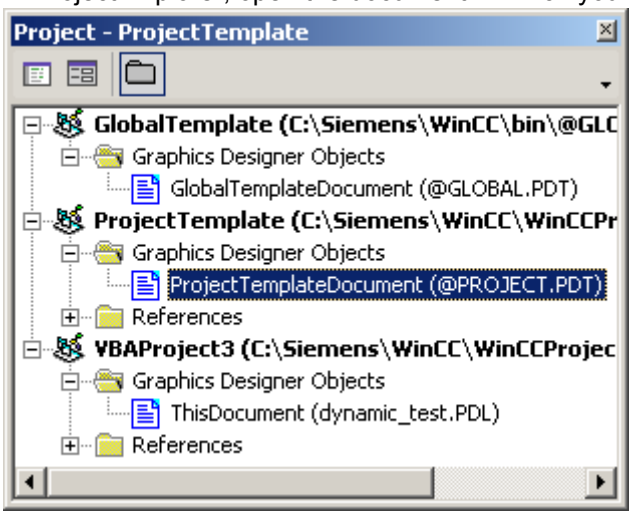
Introduction

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:

```
Sub AddNewFolderToProjectLibrary()
'VBA20
Dim objProjectLib As HMISymbolLibrary
Set objProjectLib = Application.SymbolLibraries(2)
'
'("AddFolder(DefaultName)"-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:

```
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":

```
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(DefaultName, 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 1962)

SymbolLibraries Object (Listing) (Page 1963)

PasteClipboard Method (Page 1781)

Delete Method (Page 1745)

CopyToClipboard Method (Page 1740)

AddItem Method (Page 1720)

AddFromClipboard Method (Page 1717)

AddFolder Method (Page 1716)

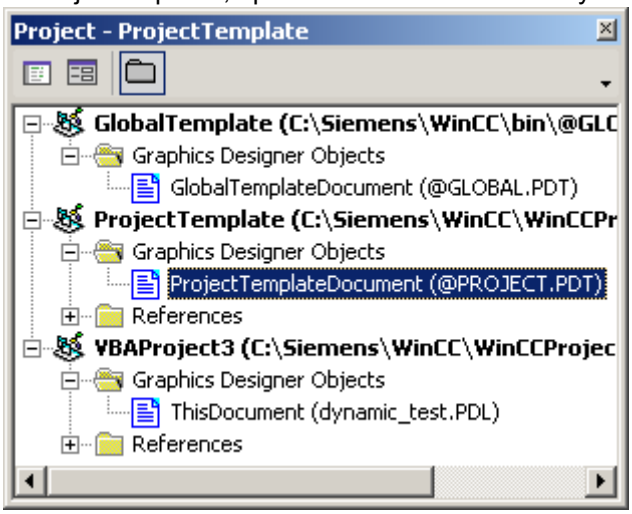
How to Insert an Object from the Component Library into a Picture with VBA (Page 1599)

Accessing the component library with VBA (Page 1593)

How to Insert an Object from the Component 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

```
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 1781)

CopyToClipboard Method (Page 1740)

How to Edit the Component Library with VBA (Page 1596)

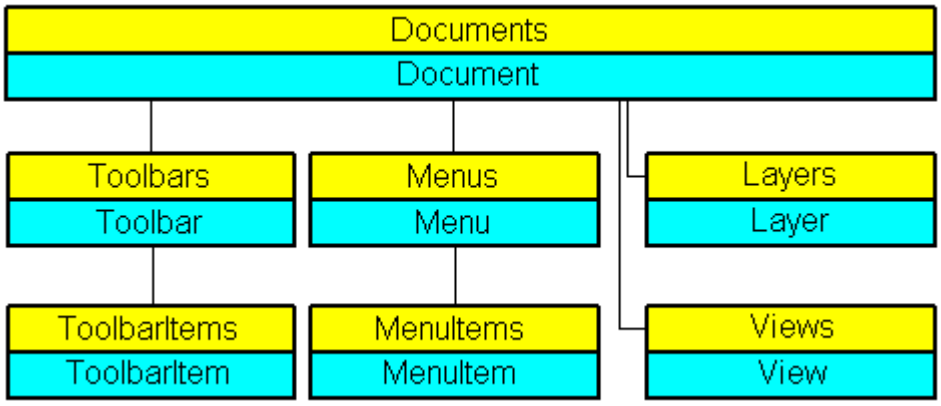
Accessing the component library with VBA (Page 1593)

4.3.3 Editing Pictures with VBA

4.3.3.1 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.



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.

See also

Editing a Copy of a Picture with VBA (Page 1605)

How to Create Picture-Specific Menus and Toolbars (Page 1602)

Editing Layers with VBA (Page 1603)

Editing Objects with VBA (Page 1606)

Adapting the Graphics Designer with VBA (Page 1569)

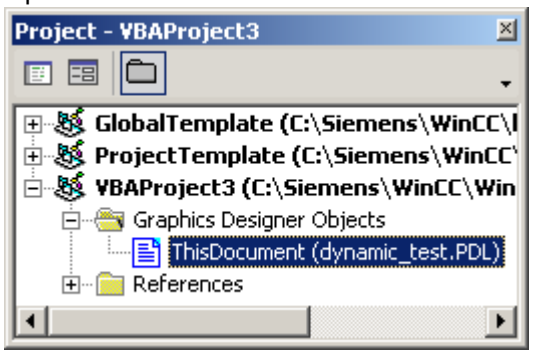
4.3.3.2 How to Create Picture-Specific Menus and Toolbars

Introduction

Picture-specific menus and toolbars are linked to a specific picture and remain visible for as long as the 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":

```
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":

```
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. Start the procedure with <F5>.

See also

Add Method (CustomToolbars Listing) (Page 1708)

InsertMenu Method (Page 1763)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Add a New Icon to the Toolbar (Page 1584)

How to Create Menus in Multiple Languages (Page 1580)

How to Add a New Menu Item to the Menu (Page 1577)

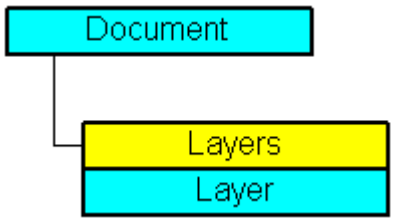
Creating Customized Menus and Toolbars (Page 1573)

Configuring Menus and Toolbars (Page 1572)

4.3.3.3 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:



In the Graphics Designer the lowest layer is "Layer 0". To give back the lowest layer with VBA, use the index "1":

```
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:

```
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.

See also

IsRTLayerVisible Method (Page 1771)

SetRTLayerVisible Method (Page 1799)

SetCSLayerVisible Method (Page 1797)

IsCSLayerVisible Method (Page 1770)

Layers Object (Listing) (Page 1895)

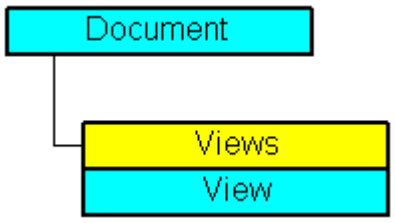
Editing Pictures with VBA (Page 1601)

Language-Dependent Configuration with VBA (Page 1570)

4.3.3.4 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:



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:

```
Sub CreateAndActivateView()
'VBA28
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Activate
End Sub
```

Editing a copy of a picture

Jede Kopie eines Bildes können Sie wie folgt bearbeiten:

- 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:

```
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
```

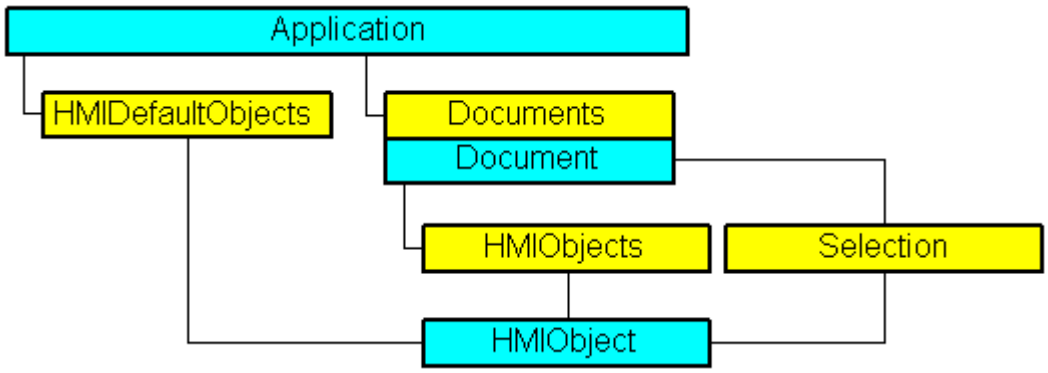
See also

Add Method (Views Listing) (Page 1713)
ScrollPosY Property (Page 2284)
ScrollPosX Property (Page 2283)
ActiveLayer Property (Page 1996)
View Object (Page 1991)
SetCSLayerVisible Method (Page 1797)
IsCSLayerVisible Method (Page 1770)
Activate Method (Page 1704)
Editing Layers with VBA (Page 1603)
Editing Pictures with VBA (Page 1601)

4.3.4 Editing Objects with VBA

4.3.4.1 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 1890)

VBA Reference (Page 1664)

Underlined Property (Page 2324)

Size Property (Page 2291)

Parent Property (Page 2241)

LanguageID Property (Page 2158)

Italic Property (Page 2151)

FontFamily Property (Page 2127)

Bold Property (Page 2034)

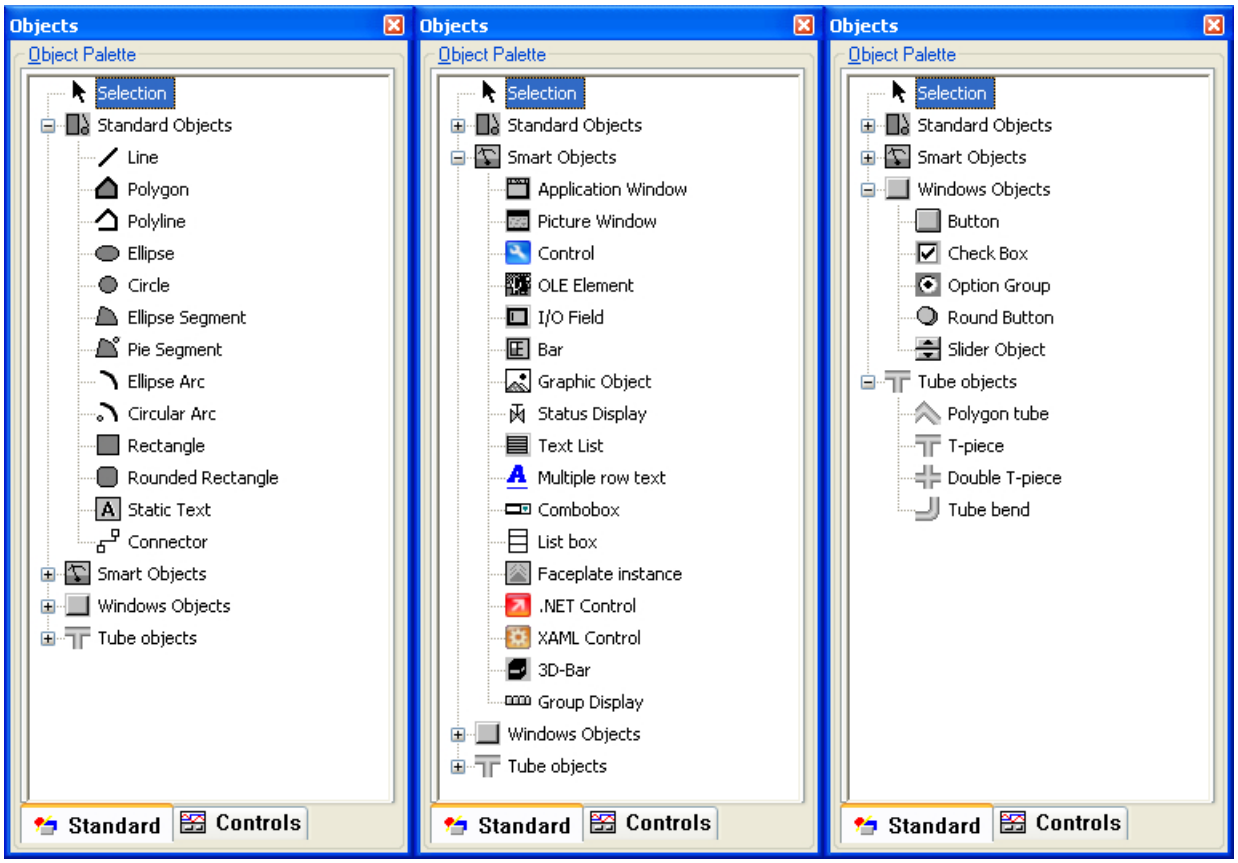
Application Property (Page 2008)

4.3.4.2 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 you will find these objects on the object palette on the "Standard" tab:



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 in 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

See also

Parent Property (Page 2241)

Item Property (Page 2152)

Count Property (Page 2082)

Application Property (Page 2008)

How to edit Default objects, Smart objects, Windows objects and Tube objects

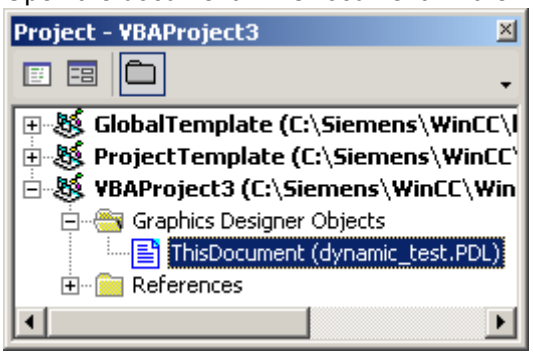
Introduction

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:

```
Sub EditDefinedObjectType()
'VBA33
Dim objCircle As HMICircle
Set objCircle =
ActiveDocument.HMIObjects.AddHMIObject("myCircleAsCircle",
"HMICircle")
With objCircle
'direct calling of objectproperties available
.BorderWidth = 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:

```
Sub EditHMIObject()
'VBA34
Dim objObject As HMIObject
Set objObject =
ActiveDocument.HMIObjects.AddHMIObject("myCircleAsObject",
"HMICircle")
With objObject
'Access to objectproperties only with property "Properties":
.Properties("BorderWidth") = 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:

```
Sub SelectObject()
'VBA35
Dim objObject As HMIObject
Set objObject =
ActiveDocument.HMIObjects.AddHMIObject("mySelectedCircle",
"HMICircle")
ActiveDocument.HMIObjects("mySelectedCircle").Selected = True
End Sub
```

6. To search for an object in the current picture, insert a "FindObjectsByName()", "FindObjectsByType()," or "FindObjectsByProperty()" 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 HMICollection
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 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 & "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 HMICollection
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.

```
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 1755)

Delete Method (Page 1745)

AddHMIObject Method (Page 1719)

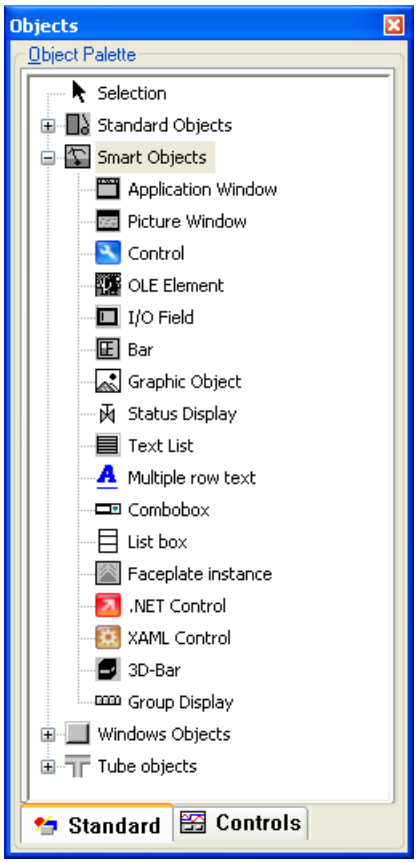
How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)

Editing Objects with VBA (Page 1606)

OLE Objects

Introduction

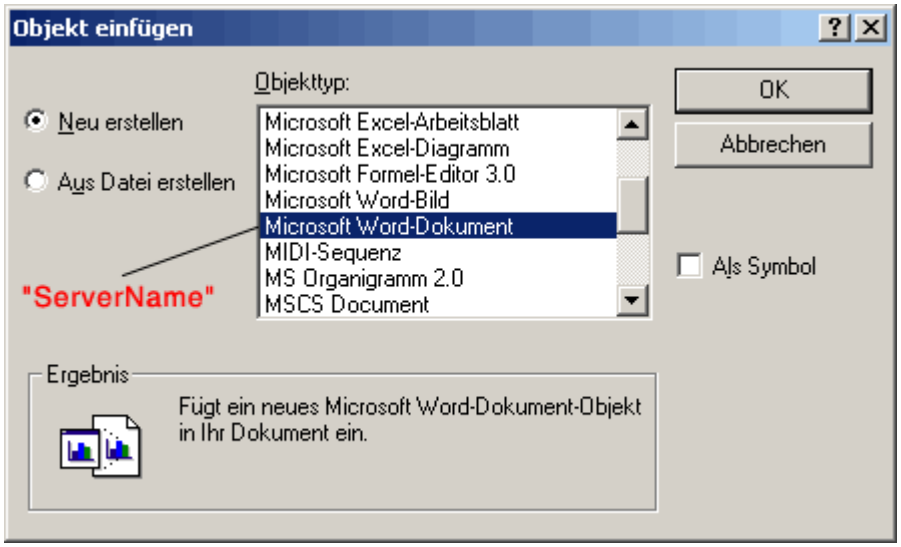
You can use VBA to insert OLE Elements into a picture. The OLE Element belongs to the Smart objects. In the Graphics Designer you will find it on the object palette on the "Standard" tab:



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:



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:

```
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 HMIObject object.

See also

OLEObject Object (Page 1914)

AddOLEObject Method (Page 1722)

How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)

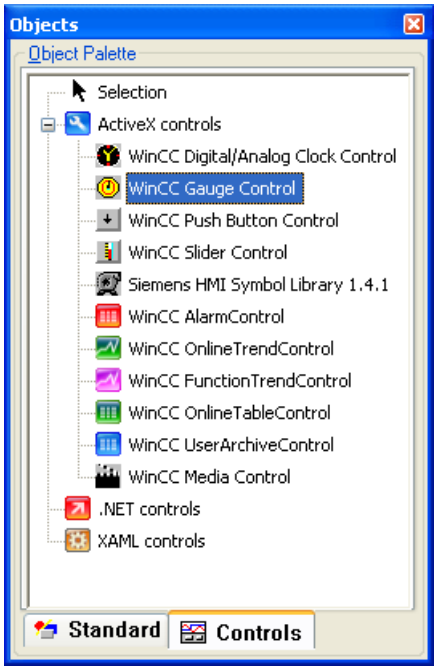
Default objects, Smart objects, Windows objects and Tube objects (Page 1609)

Editing Objects with VBA (Page 1606)

ActiveX controls

Introduction

You can use VBA to insert ActiveX controls into a picture. In the Graphics Designer you will find the ActiveX controls supplied with WinCC on the object palette on the Controls tab:



Further information is provided under "AddActiveXControl method" in this documentation and under "Object Palette" 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"):

```
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:

```
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
```

See also

ActiveXControl Object (Page 1811)

AddActiveXControl Method (Page 1714)

How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)

Default objects, Smart objects, Windows objects and Tube objects (Page 1609)

Editing Objects with VBA (Page 1606)

4.3.4.3 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:

```
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:

```
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

Selection Object (Listing) (Page 1949)

GroupedObjects Object (Listing) (Page 1877)

Ungroup Method (Page 1803)

Remove Method (Page 1784)

Delete Method (Page 1745)

Add Method (GroupedObjects Listing) (Page 1710)

How to Edit Objects in a Group Object with VBA (Page 1623)

How to Edit Group Objects with VBA (Page 1620)

Editing Objects with VBA (Page 1606)

VBA in the Graphics Designer (Page 1567)

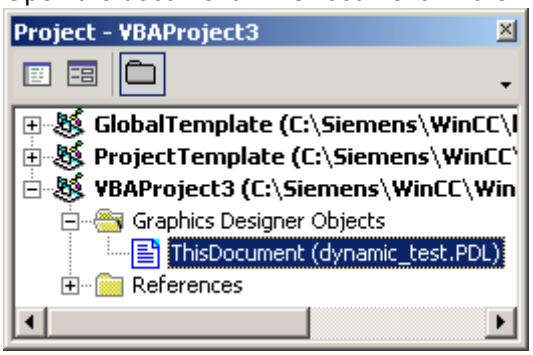
How to Edit Group Objects with VBA

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.

```
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. Start the procedure with <F5>.

See also

Ungroup Method (Page 1803)

Remove Method (Page 1784)

Delete Method (Page 1745)

CreateGroup Method (Page 1743)

Add Method (GroupedObjects Listing) (Page 1710)

Selection Object (Listing) (Page 1949)

GroupedObjects Object (Listing) (Page 1877)

Group Object (Page 1873)

How to Edit Objects in a Group Object with VBA (Page 1623)

Group Objects (Page 1618)

Editing Objects with VBA (Page 1606)

VBA in the Graphics Designer (Page 1567)

How to Edit Objects in a Group Object with VBA

Introduction

Here you will find the following instructions for editing objects in a group object with VBA:

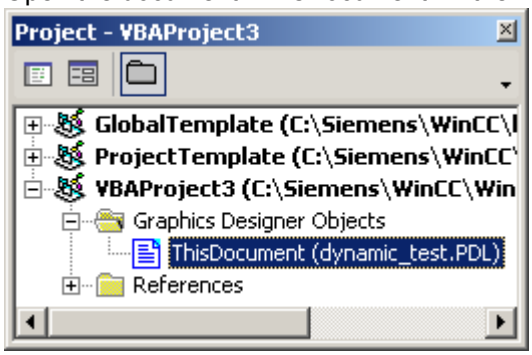
- Editing a property of an object in the group object
- Editing a property of all objects in the group object

Requirements

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":

```
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:

```
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. Start the procedure with <F5>.

See also

Properties Object (Listing) (Page 1931)

GroupedObjects Object (Listing) (Page 1877)

Ungroup Method (Page 1803)

Remove Method (Page 1784)

Delete Method (Page 1745)

Add Method (GroupedObjects Listing) (Page 1710)

How to Edit Group Objects with VBA (Page 1620)

Group Objects (Page 1618)

Editing Objects with VBA (Page 1606)

VBA in the Graphics Designer (Page 1567)

4.3.4.4 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.

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 1882)

CustomizedObject Object (Page 1839)

Destroy Method (Page 1749)

Delete Method (Page 1745)

CreateCustomizedObject Method (Page 1741)

How to Edit a Customized Object with VBA (Page 1628)

How to Edit Group Objects with VBA (Page 1620)

Group Objects (Page 1618)

Editing Objects with VBA (Page 1606)

How to Edit a Customized Object with VBA

Introduction

Here you will find the following instructions for editing a customized object with VBA:

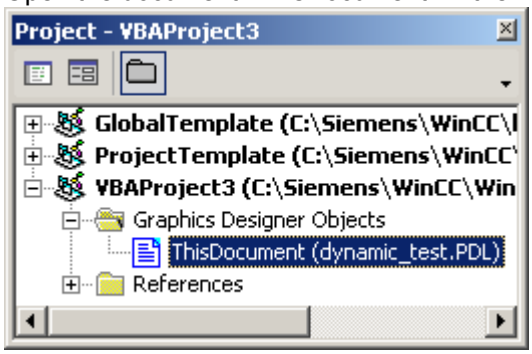
- Creating a customized object from selected objects
- Ungrouping Customized Objects
- Deleting a customized object

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:

```
Sub CreateCustomizedObject()
'VBA54
Dim objCustomizedObject As HMICustomizedObject
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 costumize-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. Start the procedure with <F5>.

See also

Destroy Method (Page 1749)

Delete Method (Page 1745)

CreateCustomizedObject Method (Page 1741)

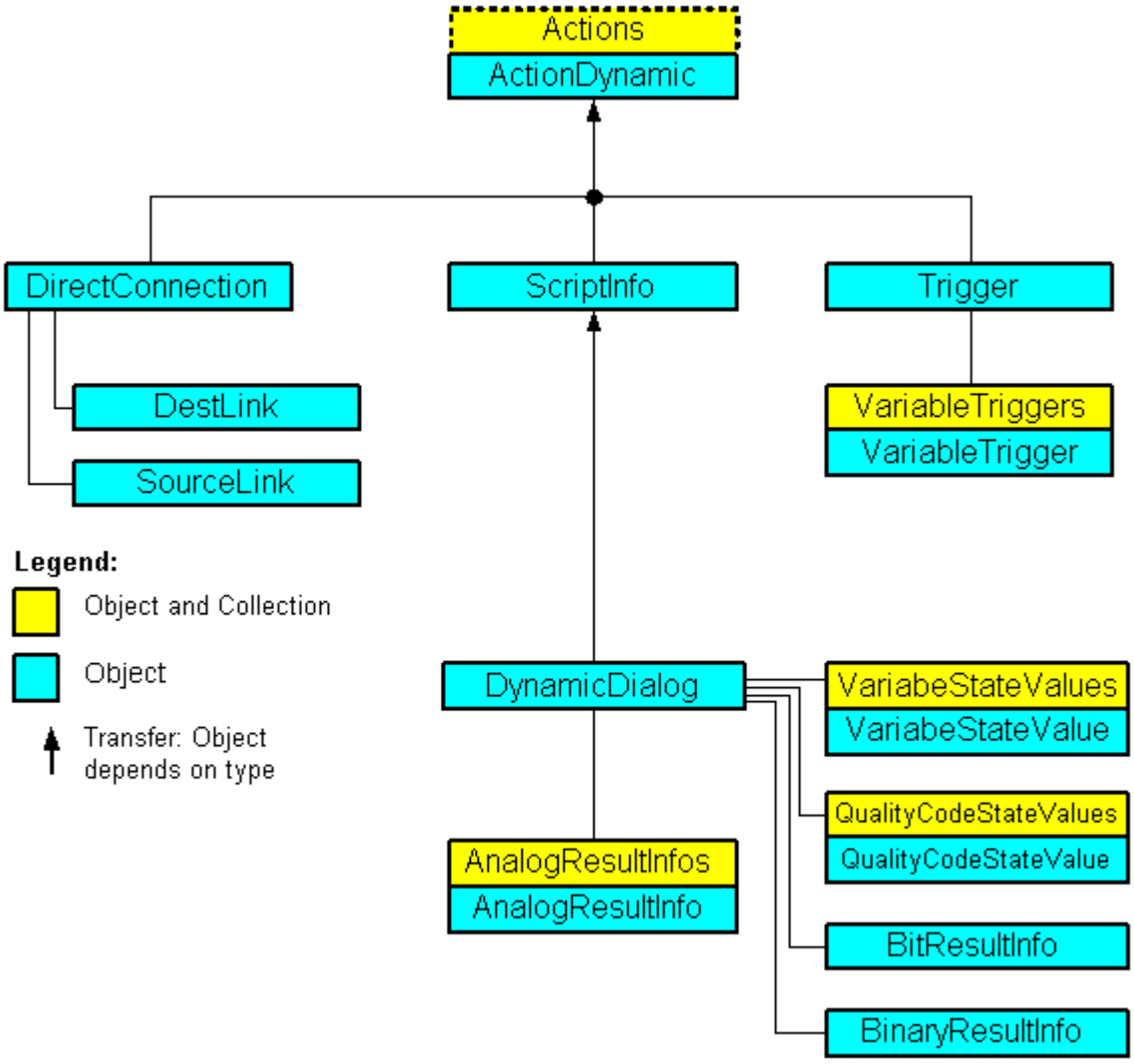
Customized Objects (Page 1626)

4.3.5 Creating Dynamics with VBA

4.3.5.1 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:



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 1652)

Configuring Event-Driven Actions with VBA (Page 1644)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

4.3.5.2 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:

```
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 1989)

VariableStateValue Object (Page 1986)

Trigger Object (Page 1974)

ScriptInfo Object (Page 1948)

BinaryResultInfo Object (Page 1822)

AnalogResultInfos Object (Listing) (Page 1813)

How to Dynamize a Property with a VB Script (Page 1642)

How to Dynamize a Property with a C Script (Page 1639)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

How to Dynamize a Property with Tag Connection (Page 1634)

Creating Dynamics with VBA (Page 1631)

How to Dynamize a Property with Tag Connection

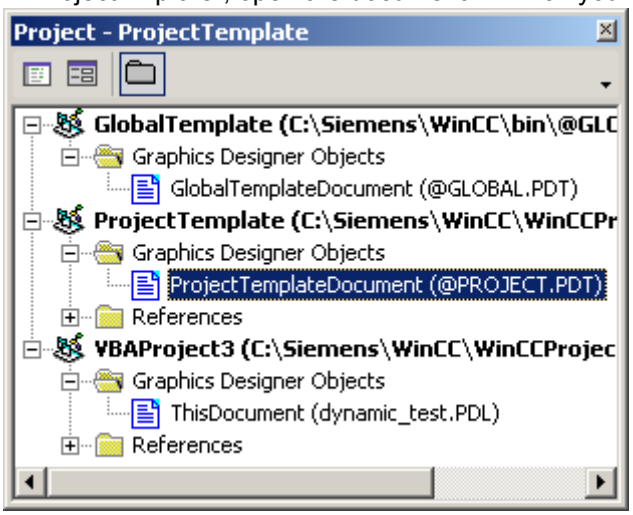
Introduction

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:



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()
'VBA58
Dim objVariableTrigger As HMIVariableTrigger
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("Circle1",
"HMICircle")
'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":

```
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(hmiDynamicCreationTypeVariableIndirec
t, "Anton")
'
'Define cycle-time
With objVariableTrigger
.CycleType = hmiVariableCycleType_2s
End With
End Sub
```

5. Start the procedure with <F5>.

See also

CycleType Property (Page 2088)

VarName Property (Page 2403)

VariableTrigger Object (Page 1989)

CreateDynamic Method (Page 1742)

How to Dynamize a Property with a VB Script (Page 1642)

How to Dynamize a Property with a C Script (Page 1639)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

Creating Dynamics with VBA (Page 1631)

How to Dynamize a Property with Dynamic Dialog

Introduction

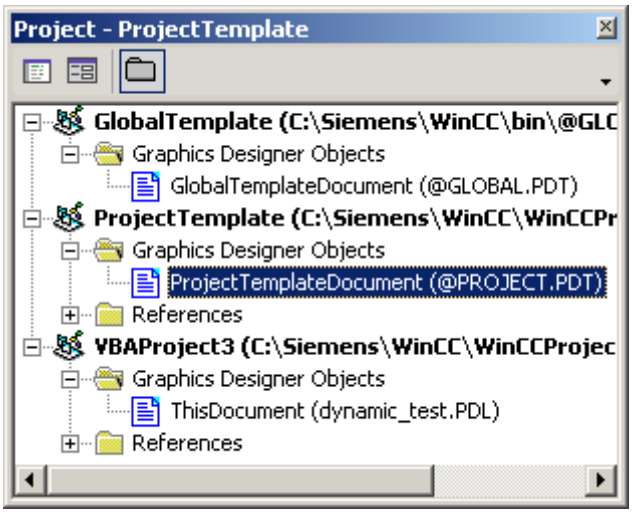
You can use the Dynamic dialog to dynamize properties of pictures and objects depending on certain value ranges or variable states. The following value ranges are available for selection:

- 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:



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 will be dynamically configured using 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
```

Initialize "ToolTipText" property

The property "ToolTipText" must be initialized before being made dynamic, by assigning a text here.

```
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 Tag Connection (Page 1634)

ResultType Property (Page 2272)

DynamicDialog Object (Page 1851)

CreateDynamic Method (Page 1742)

How to Dynamize a Property with a VB Script (Page 1642)

How to Dynamize a Property with a C Script (Page 1639)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

Creating Dynamics with VBA (Page 1631)

ToolTipText Property (Page 2310)

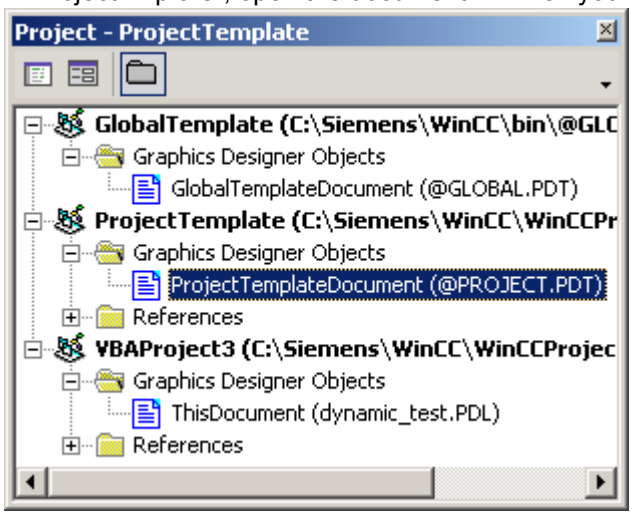
How to Dynamize a Property with a C Script

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:

```
Sub AddDynamicAsCSkriptToProperty()
'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""",lHeight);"
strCode = strCode & vbCrLf & "//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 2314)

ScriptType Property (Page 2280)

SourceCode Property (Page 2295)

CycleType Property (Page 2088)

ScriptInfo Object (Page 1948)

CreateDynamic Method (Page 1742)

How to Dynamize a Property with a VB Script (Page 1642)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

How to Dynamize a Property with Tag Connection (Page 1634)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

Creating Dynamics with VBA (Page 1631)

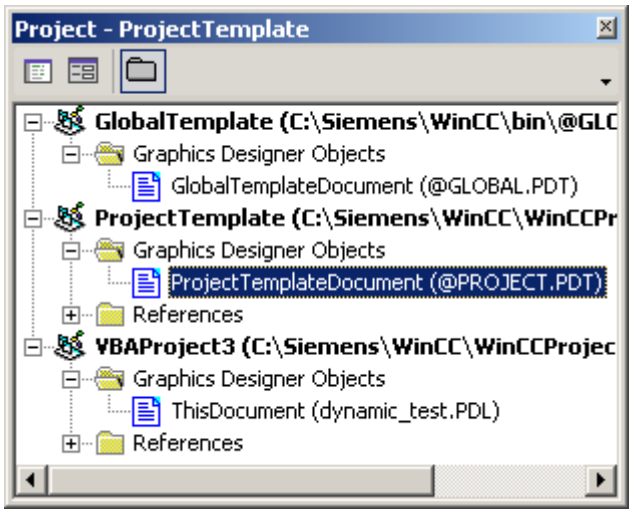
How to Dynamize a Property with a VB Script

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 AddDynamicAsVBSkriptToProperty()
'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 1639)

Trigger Property (Page 2314)

SourceCode Property (Page 2295)

CycleType Property (Page 2088)

ScriptInfo Object (Page 1948)

CreateDynamic Method (Page 1742)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

How to Dynamize a Property with Tag Connection (Page 1634)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

Creating Dynamics with VBA (Page 1631)

4.3.5.3 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:

```
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":

```
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 an Event-Oriented VB action with VBA (Page 1650)

Events Property (Page 2099)

ScriptInfo Object (Page 1948)

Event Object (Page 1862)

How to Configure an Event-Oriented C Action with VBA (Page 1648)

How to Configure a Direct Connection with VBA (Page 1645)

Event Handling (Page 1655)

Creating Dynamics with VBA (Page 1631)

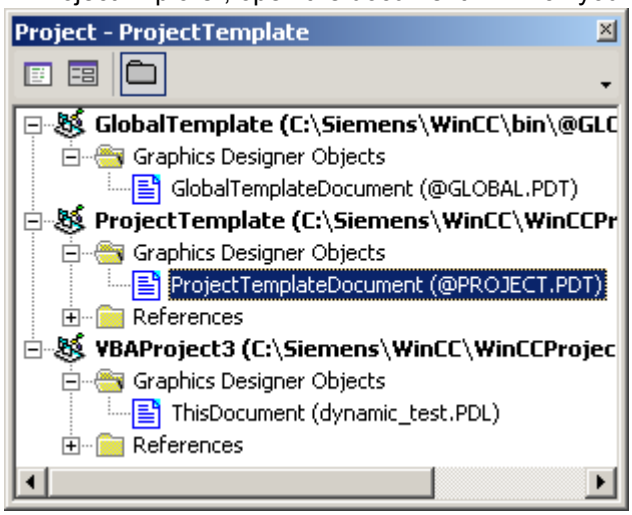
How to Configure a Direct Connection with VBA

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 HMIDirectConnection
'
'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
'
'Directconnection is initiated by mouseclick:
Set objDConnection =
objButton.Events(1).Actions.AddAction(hmiActionCreationTypeDirectC
onnection)
With objDConnection
'Sourceobject: Property "Top" of Rectangle_A
.SourceLink.Type = hmiSourceTypeProperty
.SourceLink.ObjectName = "Rectangle_A"
.SourceLink.AutomationName = "Top"
'
'Destinationobject: Property "Left" of Rectangle_B
.DestinationLink.Type = hmiDestTypeProperty
.DestinationLink.ObjectName = "Rectangle_B"
.DestinationLink.AutomationName = "Left"
End With
```

```
End Sub
```

4. Start the procedure with <F5>.

See also

ObjectName Property (Page 2232)

AutomationName Property (Page 2011)

SourceLink Object (Page 1955)

DestLink Object (Page 1843)

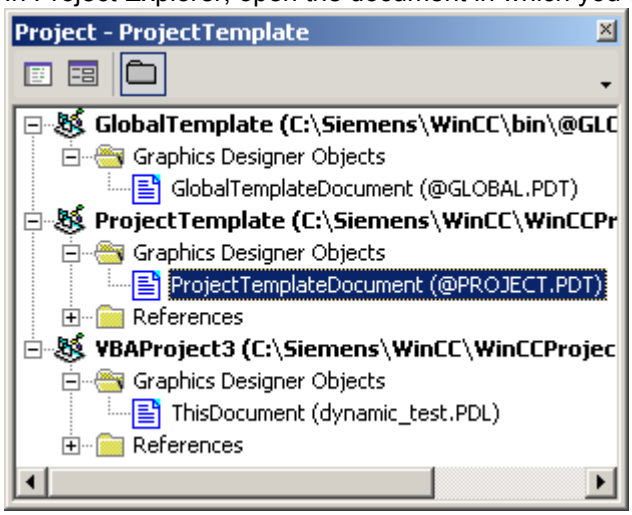
DirectConnection Object (Page 1845)

Configuring Event-Driven Actions with VBA (Page 1644)

How to Configure an Event-Oriented C Action 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 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 radius of the circle enlarges 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
strCode = strCode & "lHeight = GetHeight (""events.PDL"",
""myCircle"");"
strCode = strCode & vbCrLf & "lHeight = lHeight+5;" & vbCrLf &
"check = "
strCode = strCode & "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 Radius"
End With
'Configure directconnection:
Set objCScript =
objButton.Events(1).Actions.AddAction(hmiActionCreationTypeCScript
)
With objCScript
'
'Note: Replace "events.PDL" with your picturename
.SourceCode = strCode
End With
End Sub
```

4. Start the procedure with <F5>.

See also

ScriptInfo Object (Page 1948)

Events Object (Listing) (Page 1863)

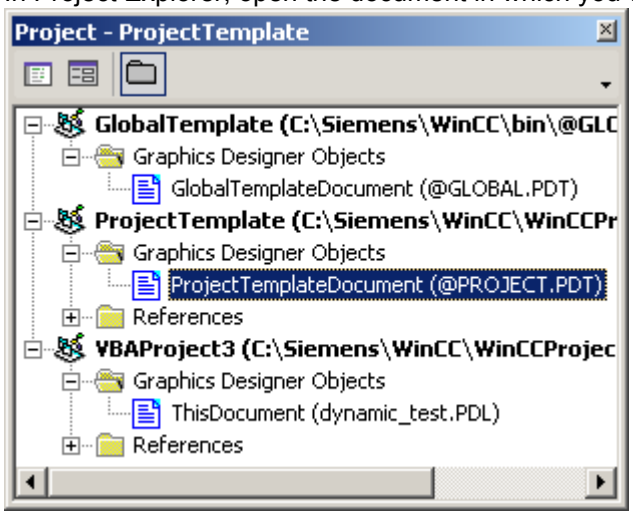
Actions Object (Listing) (Page 1810)

Configuring Event-Driven Actions with VBA (Page 1644)

How to Configure an Event-Oriented VB action 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 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:

```
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 1810)

ScriptInfo Object (Page 1948)

Events Object (Listing) (Page 1863)

Configuring Event-Driven Actions with VBA (Page 1644)

4.3.5.4 Editing Triggers

Editing Triggers

Introduction

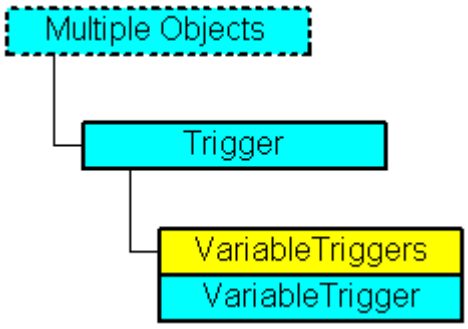
You use triggers in connection with dynamics for graphics objects and for triggering actions on object properties. Examples of triggers include:

- 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 1653)

VariableTrigger Object (Page 1989)

Trigger Object (Page 1974)

ScriptInfo Object (Page 1948)

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 1989)
Trigger Object (Page 1974)
ScriptInfo Object (Page 1948)
Editing Triggers (Page 1652)

4.3.6 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.

NOTICE
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".

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:

```
Sub Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty, CancelForwarding As
Boolean)
```

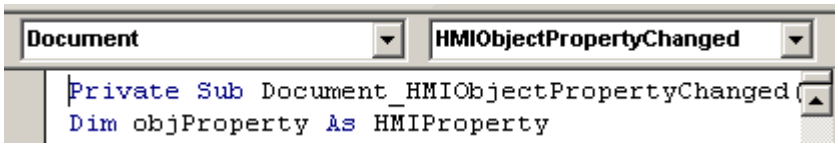
```
'VBA72
CancelForwarding = True
MsgBox "Object's property has been changed!"
End Sub
```

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:



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.

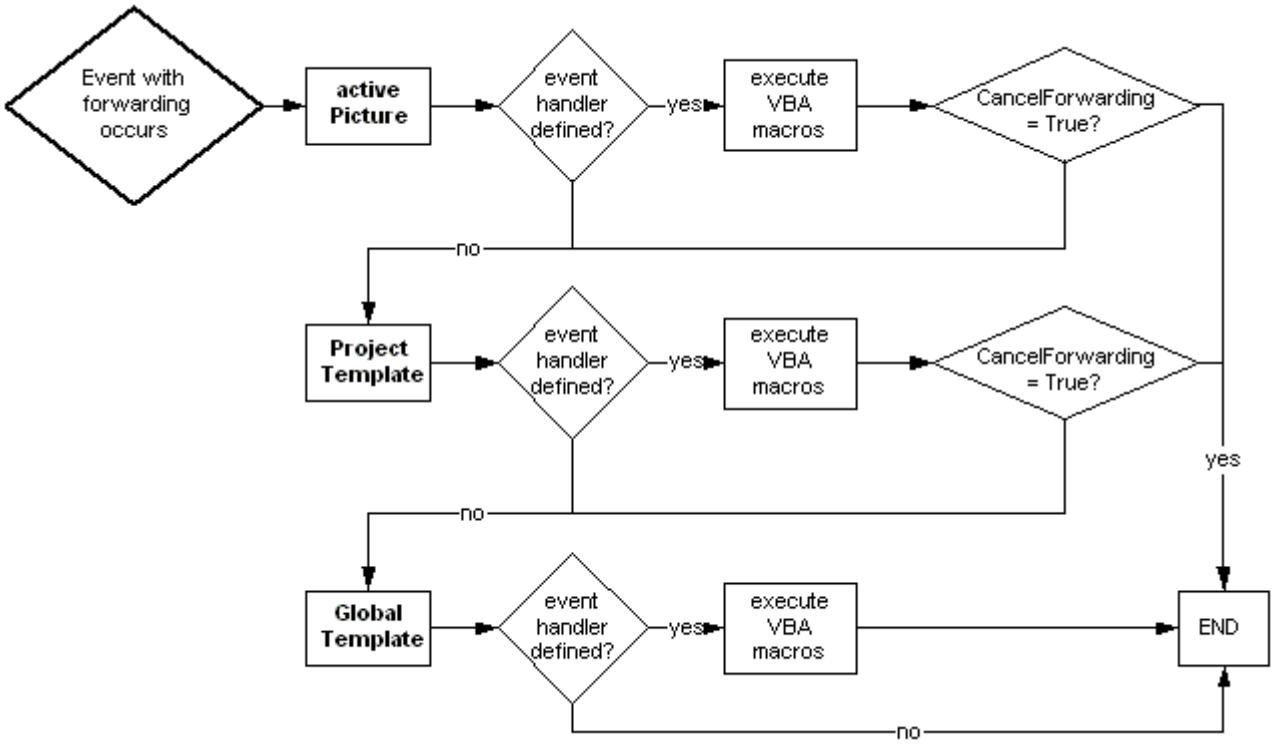
To make the application-specific events available, write the following statement in the VBA editor at the start of the document (preferably the "Project Template" or "Global Template"):

```
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:

Example 1: Occurrence of an event with forwarding

The illustration shows the sequence that follows from the occurrence of an event with forwarding:

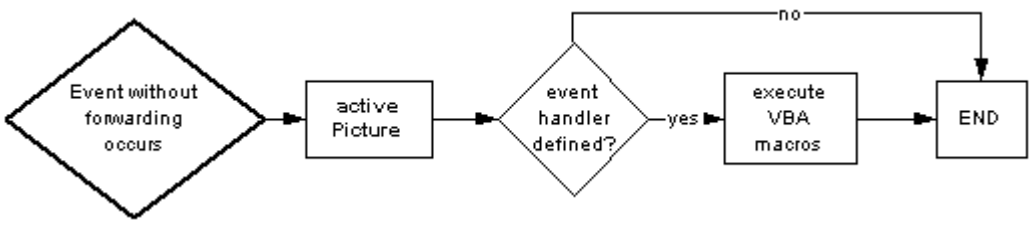


Note

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 2092)

Organizing VBA Code in a WinCC Project (Page 1562)

4.3.7 Accessing External Applications with VBA

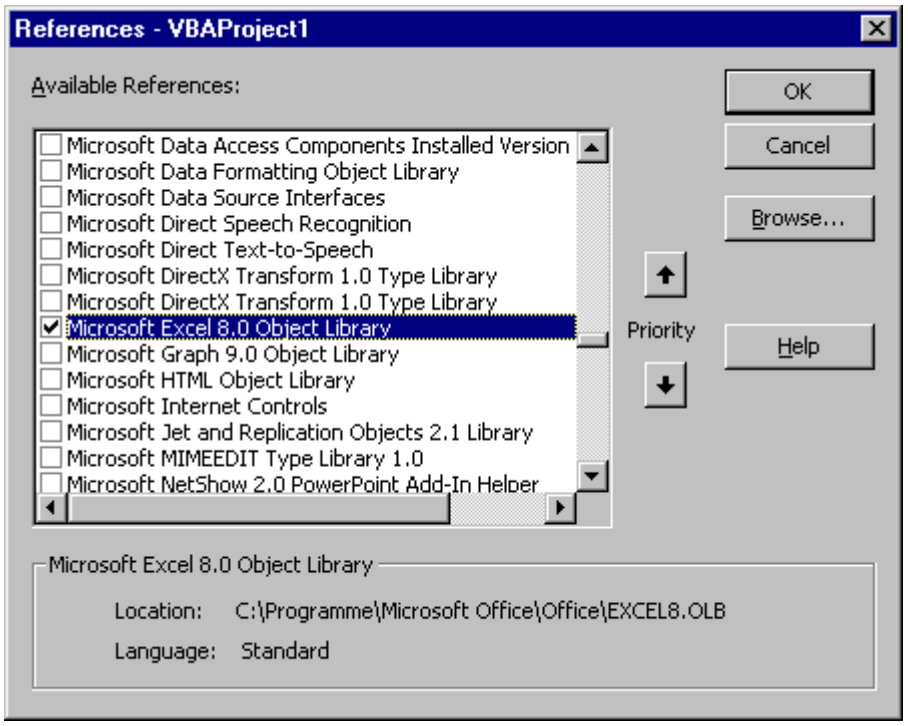
4.3.7.1 Accessing External Applications with VBA

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.

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:



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 1659)

4.3.7.2 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.

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 = Application
Set objXLS = New Excel.Application

'Create workbook
Set objWBook = objXLS.Workbooks.Add()
objWBook.SaveAs objGDApplication.ApplicationDataPath & "DefaultObjekte.xls"

'Adds 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
'Select and groups the range of object-properties in the worksheet
Set rngSelection = objWSheet.Range(objWSheet.Rows(lRowGroupStart), _
                    objWSheet.Rows(lRow - 1))
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 objProperty 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
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 grafexe.Application
Dim objDoc As grafexe.Document
Dim objHMIObject As grafexe.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 cancled!", 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:" & vbCrLf & strSheets, "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)
Else
```

```
    Set objHMIObject = objDoc.HMIObjects.AddHMIObject(.Cells(lRow, 1).value, .Cells(lRow,
2).value)
  End If
  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
```

4.4 The object model of the Graphics Designer

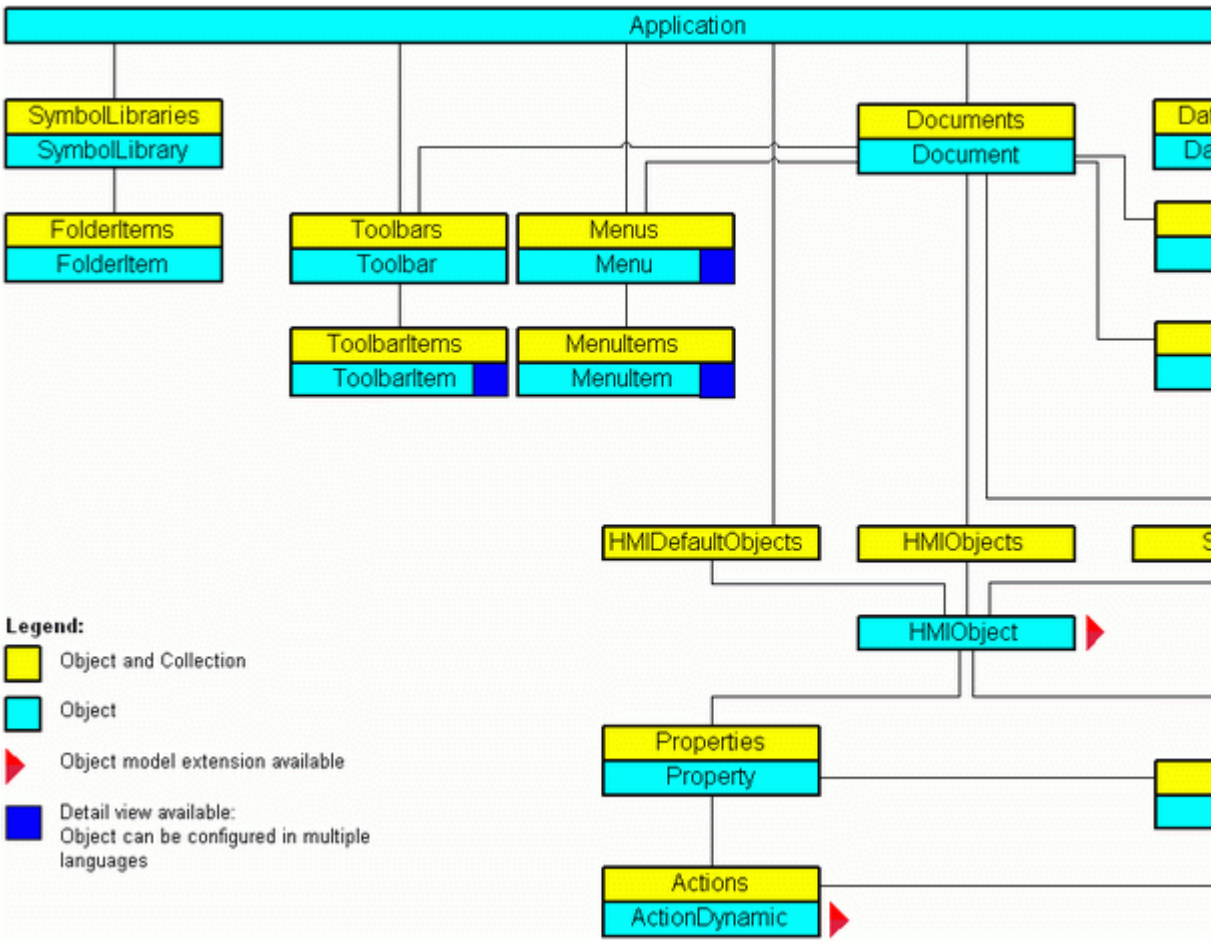
4.4.1 VBA Reference

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. "HMISymbolLibrary".


Application
SymbolLibraries
SymbolLibrary
FolderItems
FolderItem
Toolbars
Toolbar
Menus
Menu
ToolbarItems
ToolbarItem
MenuItems
MenuItem
Documents
Document
HMIDefaultObjects
HMIObjects
HMIObject
Properties
Property
Actions
ActionDynamic
Legend:
Object and Collection
Object
Object model extension available
Detail view available:
Object can be configured in multiple languages

See also

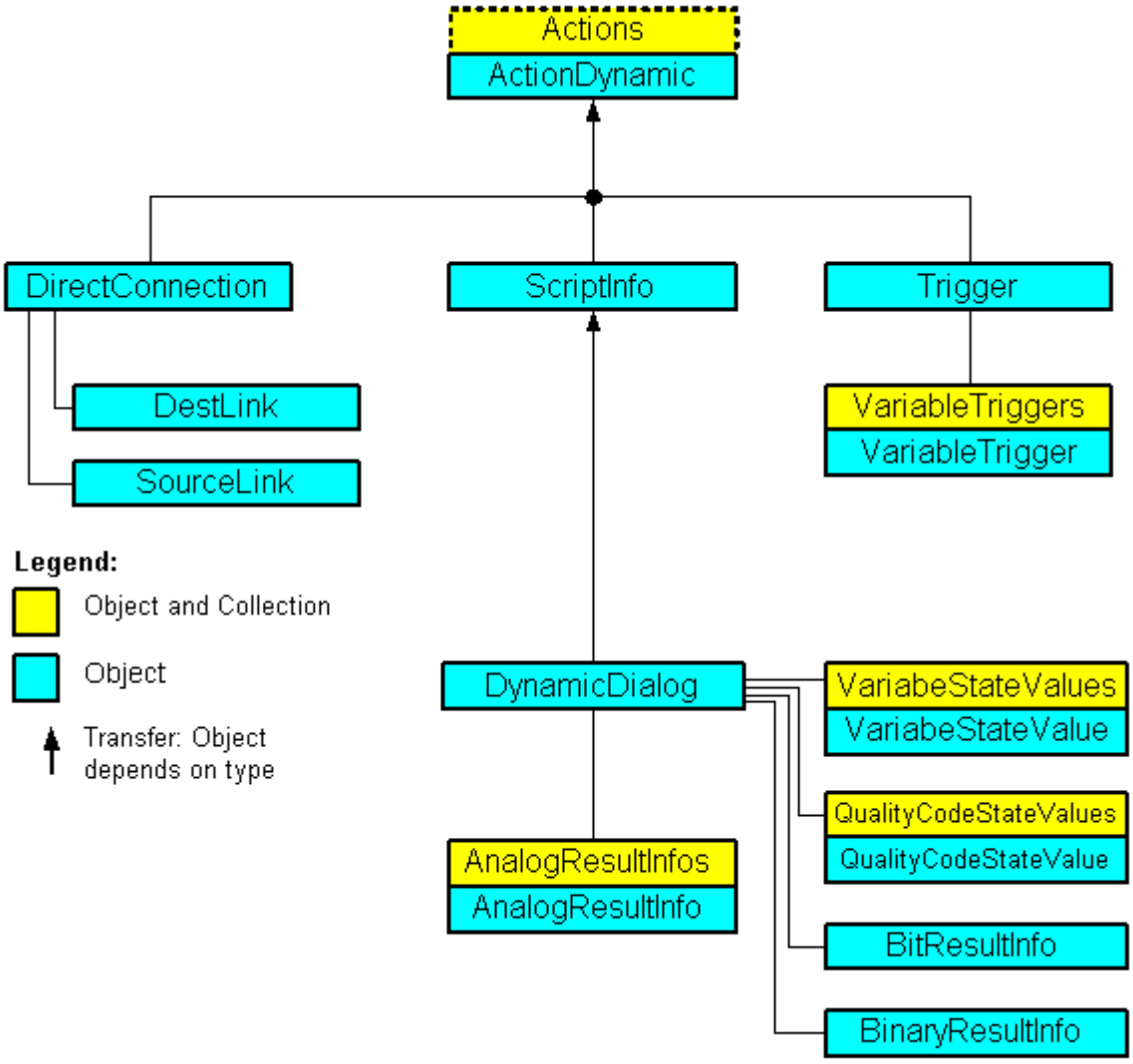
Events Object (Listing) (Page 1863)
SymbolLibraries Object (Listing) (Page 1963)
Actions Object (Listing) (Page 1810)
Application Object (Page 1814)
DataLanguage Object (Page 1841)
DataLanguages Object (Listing) (Page 1842)
Document Object (Page 1847)
Documents Object (Listing) (Page 1850)
Event Object (Page 1862)
HMIDefaultObjects Object (Listing) (Page 1878)
HMIObject Object (Page 1882)
HMIObjects Object (Listing) (Page 1884)
FolderItem Object (Page 1867)
FolderItems Object (Listing) (Page 1868)
VBA Reference: ActionDynamic (Page 1667)
VBA Reference: HMIObjects (Page 1669)
VBA Reference: Languages (Page 1671)
Layer Object (Page 1894)
Layers Object (Listing) (Page 1895)
Menu Object (Page 1903)
Menus Object (Listing) (Page 1904)
MenuItem Object (Page 1906)
MenuItems Object (Listing) (Page 1909)
Properties Object (Listing) (Page 1931)
Toolbar Object (Page 1967)
Toolbars Object (Listing) (Page 1968)
ToolbarItem Object (Page 1970)
ToolbarItems Object (Listing) (Page 1973)
View Object (Page 1991)
Views Object (Listing) (Page 1993)
Selection Object (Listing) (Page 1949)
SymbolLibrary Object (Page 1962)
Property Object (Page 1932)

4.4.2 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.



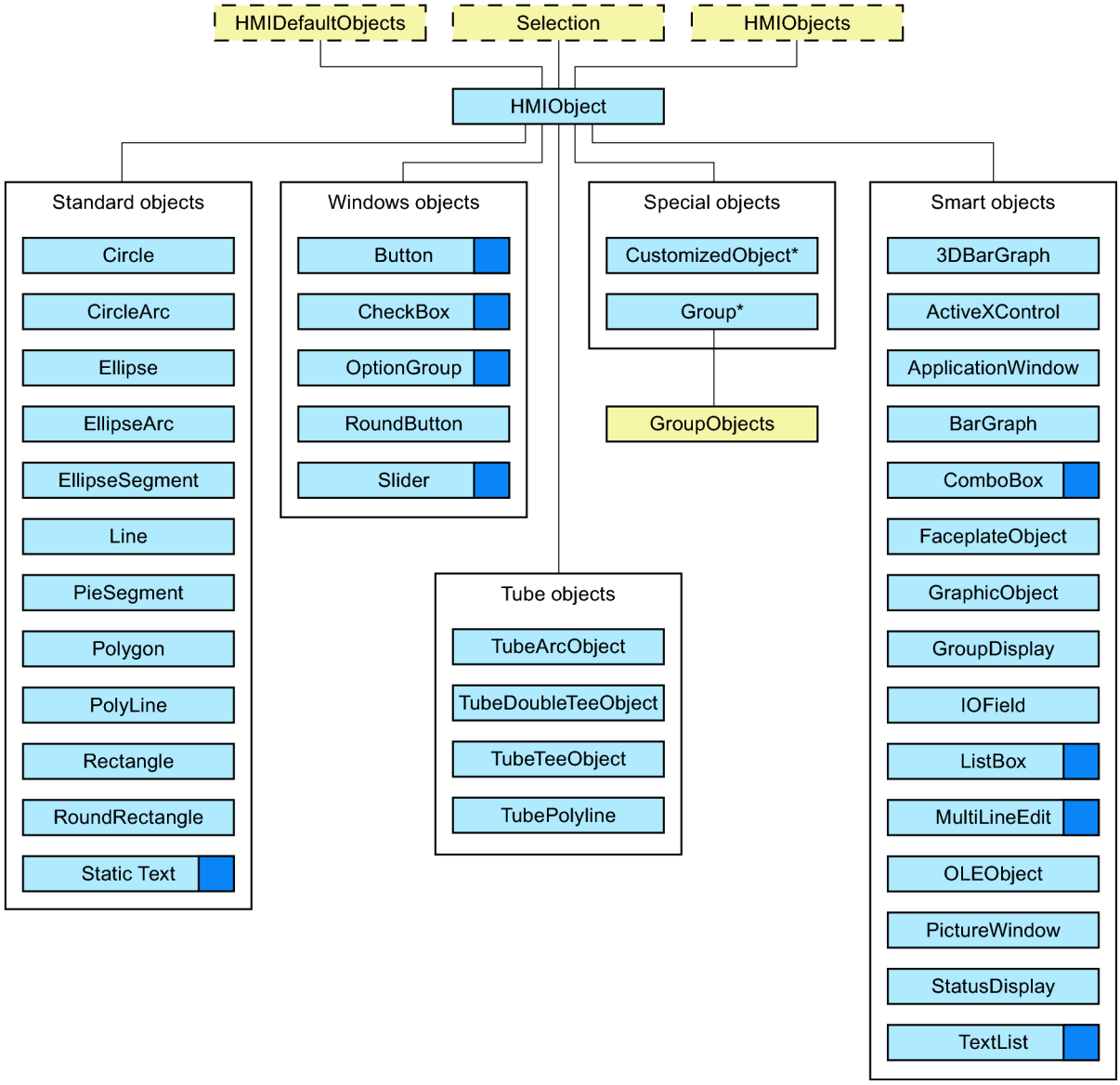
See also

VBA Reference (Page 1664)
AnalogResultInfo Object (Page 1812)
AnalogResultInfos Object (Listing) (Page 1813)
BinaryResultInfo Object (Page 1822)
BitResultInfo Object (Page 1823)
Actions Object (Listing) (Page 1810)
DestLink Object (Page 1843)
DirectConnection Object (Page 1845)
DynamicDialog Object (Page 1851)
QualityCodeStateValue Object (Page 1934)
QualityCodeStateValues Object (Listing) (Page 1936)
ScriptInfo Object (Page 1948)
SourceLink Object (Page 1955)
Trigger Object (Page 1974)
VariableStateValue Object (Page 1986)
VariableStateValues Object (Listing) (Page 1987)
VariableTrigger Object (Page 1989)
VariableTriggers Object (Listing) (Page 1990)

4.4.3 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.
Object can be configured multilingually.

* Not in DefaultObjects list.

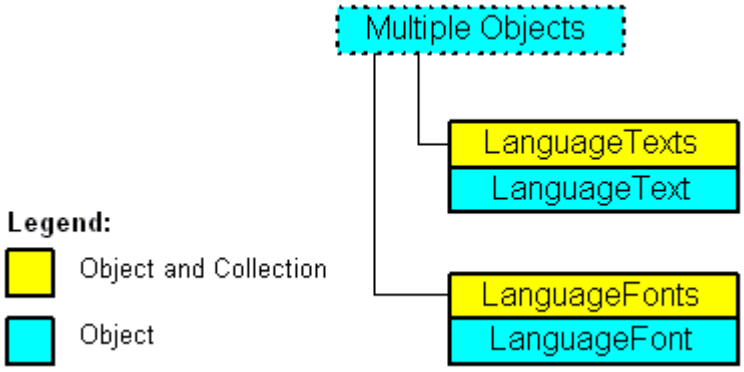
See also

VBA Reference (Page 1664)
PolyLine Object (Page 1928)
GroupDisplay Object (Page 1874)
3DBarGraph Object (Page 1805)
ActiveXControl Object (Page 1811)
ApplicationWindow Object (Page 1817)
Button Object (Page 1824)
CheckBox Object (Page 1827)
Circle Object (Page 1828)
CircularArc Object (Page 1831)
Line Object (Page 1897)
OLEObject Object (Page 1914)
OptionGroup Object (Page 1916)
PictureWindow Object (Page 1919)
PieSegment Object (Page 1922)
Polygon Object (Page 1925)
Property Object (Page 1932)
Rectangle Object (Page 1939)
RoundButton Object (Page 1942)
RoundRectangle Object (Page 1945)
Slider object (Page 1952)
StaticText Object (Page 1956)
StatusDisplay Object (Page 1959)
TextList Object (Page 1964)
Ellipse Object (Page 1853)
EllipseArc Object (Page 1856)
EllipseSegment Object (Page 1859)
GraphicObject Object (Page 1870)
Group Object (Page 1873)
HMIDefaultObjects Object (Listing) (Page 1878)
HMIObject Object (Page 1882)
HMIObjects Object (Listing) (Page 1884)
IOField Object (Page 1886)
BarGraph Object (Page 1819)
GroupedObjects Object (Listing) (Page 1877)
VBA Reference: Languages (Page 1671)
Selection Object (Listing) (Page 1949)
CustomizedObject Object (Page 1839)

4.4.4 VBA Reference: Languages

VBA Object Model: Languages

When you click an object name, you are shown a detailed description.



See also

VBA Reference (Page 1664)

LanguageFont Object (Page 1889)

LanguageFonts Object (Listing) (Page 1890)

LanguageText Object (Page 1892)

LanguageTexts Object (Listing) (Page 1893)

4.4.5 Events

4.4.5.1 A-D

Activated event

Description

Occurs 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

Parameter (Data Type)	Description
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

Event Handling (Page 1655)

BeforeClose Events

Description

Occurs immediately before a picture is closed.

syntax

```
Document_BeforeClose(Cancel As Boolean, CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
Cancel (Boolean)	TRUE if command processing is to be canceled.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

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

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that is going to be closed.
Cancel (Boolean)	TRUE if command processing is to be canceled.

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_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 1664)

BeforeDocumentSave event

Description

Occurs immediately before the picture is saved.

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_BeforeDocumentSave(Document As HMIDocument, Cancel
As Boolean)
```

Parameters

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that is going to be closed.
Cancel (Boolean)	TRUE if command processing is to be canceled.

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 1664)

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

Parameter (Data Type)	Description
HMIObject (IHMIObject)	Identifies the object to be deleted.
Cancel (Boolean)	TRUE if command processing is to be canceled.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output identifying the object to be deleted:

```
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 1664)

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:

Dim WithEvents <Name> As grafexe.Application

In the following example, the name "objGDApplication" is substituted for <Name>.

```
objGDApplication_BeforeLibraryFolderDelete(LibObject As
HMIFolderItem, Cancel As Boolean)
```

Parameter (Optional)

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The folder that is going to be deleted.
Cancel (Boolean)	TRUE if command processing is to be canceled.

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_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 1664)

BeforeLibraryObjectDelete event

Description

Occurs immediately before an object 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:

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)

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The object that is going to be deleted.
Cancel (Boolean)	TRUE if command processing is to be canceled.

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 1664)

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

Parameter (Data Type)	Description
Cancel (Boolean)	TRUE if command processing is to be canceled.

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 1664)

BeforeSave Event

Description

Occurs immediately before a picture is saved.

syntax

```
Document_BeforeSave(Cancel As Boolean, CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
Cancel (Boolean)	TRUE if command processing is to be canceled.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

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

Parameter (Data Type)	Description
Cancel (Boolean)	TRUE if command processing is to be canceled.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

--

See also

VBA Reference (Page 1664)

ConnectionEvent Event

Description

Occurs when two objects are connected via the connector.

syntax

```
ConnectionEvent(eConnEventType, HMIConnector, HMIConnectedObject,
CancelProcess, CancelForwarding)
```

Parameter (Optional)

Parameter (Data Type)	Description
eConnEventType (HMIConnectionEventType)	--
HMIConnector (HMIObject)	--
HMIConnectedObject (HMIObject)	--
CancelProcess (Boolean)	TRUE if command processing is to be canceled.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

--

See also

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
lCID (Long)	The project language identifier

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 1570)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
lCID (Long)	The user interface language identifier

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 1664)

Language-Dependent Configuration with VBA (Page 1570)

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

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that is to be activated.

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 1664)

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

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that has been created.

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 name of the newly created picture is output:

```
Private Sub objGDApplication_DocumentCreated(ByVal Document As IHMIDocument)
'VBA90
MsgBox Document.Name & " will be created."
End Sub
```

See also

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that has been opened.

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 1664)

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 HMIDocument)
```

Parameters

Parameter (Data Type)	Description
Document (HMIDocument)	The picture that has been saved.

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 saved:

```
Private Sub objGDApplication_DocumentSaved(ByVal Document As IHMIDocument)
'VBA92
MsgBox Document.Name & " is saved."
End Sub
```

See also

VBA Reference (Page 1664)

DocumentPropertyChanged event

Description

Occurs when a picture property is changed.

syntax

```
Document_DocumentPropertyChanged(ByVal Property As IHMIProperty,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
Property (IHMIProperty)	Identifies the changed property.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output identifying the picture property being changed:

```
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 1664)

4.4.5.2 F-Z

HMIObjectAdded Event

Description

Occurs when an object is added.

syntax

```
Document_HMIObjectAdded(ByVal HMIObject As IHMIObject,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
HMIObject (IHMIObject)	Identifies the object being added.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

HMIObjectMoved Event

Description

Occurs when an object is moved.

syntax

```
Document_HMIObjectMoved(ByVal HMIObject As IHMIObject,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
HMIObject (IHMIObject)	Identifies the object being moved.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

HMIObjectPropertyChanged Event

Description

Occurs when an object property is changed.

syntax

```
Document_HMIObjectPropertyChanged(ByVal Property As IHMIProperty,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
Property (IHMIProperty)	Identifies the changed property.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output identifying the object property that has been changed:

```
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 1664)

HMIObjectResized Event

Description

Occurs when the size of an object is changed.

syntax

```
Document_HMIObjectResized(ByVal HMIObject As IHMIObject,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
HMIObject (IHMIObject)	Identifies the object that is being resized.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 1664)

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

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The renamed folder.
OldName (String)	The original name of the renamed folder.

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 1664)

Accessing the component library with VBA (Page 1593)

LibraryObjectRenamed Event

Description

Occurs when an object 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_LibraryObjectRenamed(LibObject As HMIFolderItem,
OldName As String)
```

Parameters

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The renamed object.
OldName (String)	The original name of the renamed object.

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 object names are output:

```
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 1664)

Accessing the component library with VBA (Page 1593)

LibraryObjectAdded Event

Description

Occurs when an object has been added to the components library.

syntax

```
HMIObjectPropertyChanged(ByVal Property As IHMIProperty,
CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
LibObject (IHMIFolderItem)	Identifies the library object.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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 IHMIFolderItem,
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 1664)

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 IHMIMenuItem)
```

Parameters

Parameter (Data Type)	Description
MenuItem (IHMIMenuItem)	Identifies the user-defined menu.

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 entry in a user-defined menu is clicked:

```
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 userdefined 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 1589)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The newly created folder.

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 1664)

Accessing the component library with VBA (Page 1593)

NewLibraryObject Event

Description

Occurs when an object 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_NewLibraryObject(LibObject As HMIFolderItem)
```

Parameters

Parameter (Data Type)	Description
LibObject (HMIFolderItem)	The newly created object.

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 1664)

Accessing the component library with VBA (Page 1593)

Opened Event

Description

Occurs when a picture is opened.

syntax

```
Document_Opened(CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output when the picture is opened:

```
Private Sub Document_Opened(CancelForwarding As Boolean)
'VBA104
```

```
MsgBox "The Document is open now..."
End Sub
```

See also

VBA Reference (Page 1664)

Saved Event

Description

Occurs after a picture has been saved.

syntax

```
Document_Saved(CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output when the picture has been saved:

```
Private Sub Document_Saved(CancelForwarding As Boolean)
'VBA105
MsgBox "The document is saved..."
End Sub
```

See also

VBA Reference (Page 1664)

SelectionChanged Event

Description

Occurs when the selection has been changed.

syntax

```
Document_SelectionChanged(CancelForwarding As Boolean)
```

Parameters

Parameter (Data Type)	Description
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

Example:

In the following example a message is output when a new object has been selected:

```
Private Sub Document_SelectionChanged(CancelForwarding As Boolean)
'VBA106
MsgBox "The selection is changed..."
End Sub
```

See also

VBA Reference (Page 1664)

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 1664)

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

Parameter (Data Type)	Description
ToolbarItem (IHMIToolbarItem)	Identifies the symbol.

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:

```
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 1589)
VBA Reference (Page 1664)

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:

Dim WithEvents <Name> As grafexe.Application

syntax

```
Document_ViewCreated(ByVal pView As IHMIView, CancelForwarding As
Boolean)
```

Parameters

Parameter (Data Type)	Description
pView (IHMIView)	Identifies the copy of the picture.
CancelForwarding (Boolean)	TRUE if the event is not intended to be forwarded. Default setting is "False".

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
iViewCount = pView.Application.ActiveDocument.Views.Count
MsgBox "A new copy of the picture (number " & iViewCount & ") was created."
End Sub
```

See also

VBA Reference (Page 1664)

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 1664)

4.4.6 Methods

4.4.6.1 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:

```
Sub CreateAndActivateView()
'VBA111
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Activate
End Sub
```

See also

View Object (Page 1991)

Application Object (Page 1814)

VBA Reference (Page 1664)

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".

Listing	Application for the Add Method
AnalogResultInfos Listing	Adds a new, analog value range in the Dynamic dialog.
Documents Listing	Creates a new picture in the Graphics Designer
GroupedObjects Listing	Adds a new object to a group object.
Toolbars Listing	Creates a new, user-defined toolbar.
Tag Triggers Listing	Creates a new tag trigger.
Views Listing	Creates a copy of the specified picture.

See also

Add Method (Views Listing) (Page 1713)

Add Method (TagTriggers Listing) (Page 1712)

Add Method (CustomToolbars Listing) (Page 1708)

Add Method (GroupedObjects Listing) (Page 1710)

Add Method (Documents Listing) (Page 1709)

Add Method (AnalogResultInfos Listing) (Page 1706)

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

Parameter (Data Type)	Description
RangeTo (Variant)	The value range to which the change of property gives rise.
ResultValue (Variant)	The value to which the object property is assigned when the value range is reached.

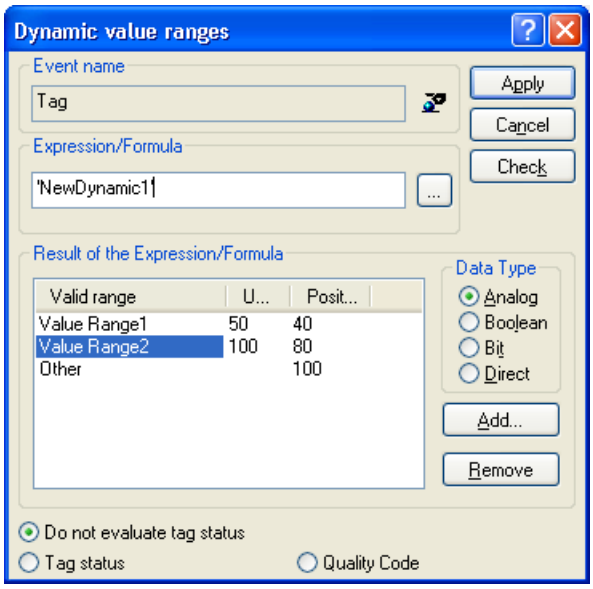
Example:

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:

```
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 1851)

AnalogResultInfos Object (Listing) (Page 1813)

CreateDynamic Method (Page 1742)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

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

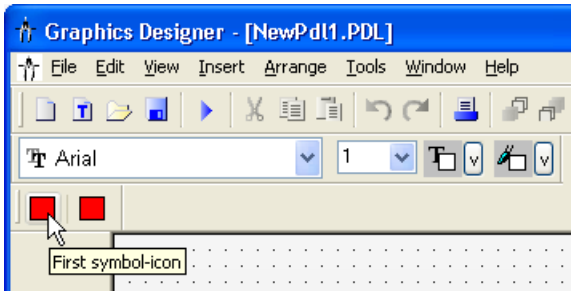
Parameter (Data Type)	Description
Key (Variant)	Identifies the user-defined toolbar. Use unique names for "Key" (e.g. "DocToolbar1")

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()
'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 1968)

InsertToolbarItem Method (Page 1769)

InsertSeparator Method (Page 1766)

InsertFromMenuItem Method (Page 1762)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

Parameter (Data Type)	Description
HMIOpenDocumentType (HMIDocumentType)	Defines how the picture will be opened: • HMIDocumentTypeVisible: Opens the picture for direct processing. This is the default setting if you do not specify the parameter. • 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.

Example:

In the following example a new picture is created in the Graphics Designer:

```
Sub AddNewDocument()
'VBA113
Application.Documents.Add hmiOpenDocumentTypeVisible
End Sub
```

See also

Hide Property (Page 2140)

Documents Object (Listing) (Page 1850)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
Index (Variant)	The object that is intended to be added. You can either use the index number or the object name.

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 1877)

Add Method (TagTriggers Listing)

Description

Creates a new tag trigger.

syntax

Expression.`Add(VarName, Type)`

Expression

Necessary. An expression or element which returns an object of the "TagTriggers" type.

Parameters

Parameter (Data Type)	Description
VarName (String)	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.
Type (CycleType)	This is the cycle type. Select the cycle type from a list in the VBA Editor when you use this method.

Example:

In the following example the radius of a circle is made dynamic using a trigger 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
```

See also

VariableTriggers Object (Listing) (Page 1990)

VBA Reference (Page 1664)

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 1993)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
HMIActionCreationType (Variant)	Defines the action: • hmiActionCreationTypeCScript: Configures a C action • hmiActionCreationTypeVBScript: Configures a VBS action • hmiActionCreationTypeDirectConnection: Configures a direct connection

Example:

In the following example a VBS action for changing the radius of a circle is configured:

```
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 1862)

Actions Object (Listing) (Page 1810)

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

Expression.`AddActiveXControl("ObjectName", "ProgID")`

Expression

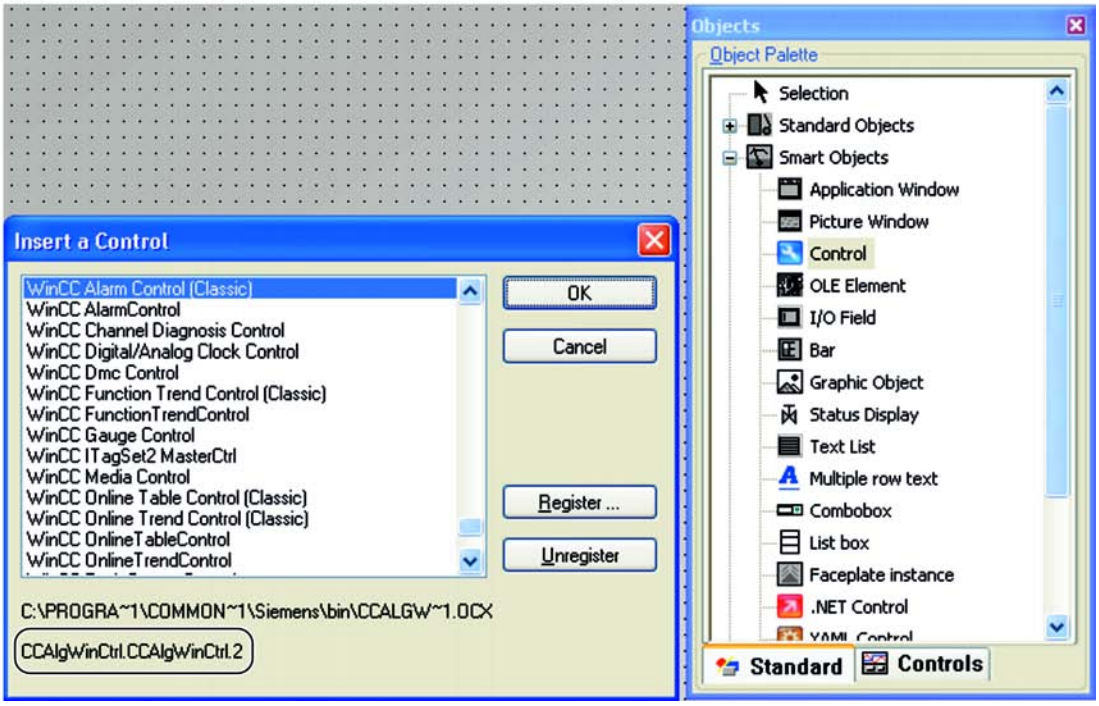
Required. An expression or element which returns an object of the "HMIObjects" type.

Parameter

Parameter (data type)	Description
ObjectName (String)	The name of the object. You can address the object by its name in a listing.
ProgID (String)	The ActiveX Control that is to be inserted.

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:

Name of the WinCC control	ProgID
Siemens HMI Symbol Library	SiemensHMI.SymbolLibrary.1
WinCC AlarmControl	CCAxAlarmControl.AxAlarmControl.1
WinCC digital/analog clock control	DACLOCK.DaclockCtrl.1
WinCC FunctionTrendControl	CCAxFunctionTrendControl.AxFunctionTrendControl.1
WinCC gauge control	XGAUGE.XGaugeCtrl.1

Name of the WinCC control	ProgID
WinCC media control	CCMediaControl.CCMediaControl.1
WinCC OnlineTableControl	CCAxOnlineTableControl.AxOnlineTableControl.1
WinCC OnlineTrendControl	CCAxOnlineTrendControl.AxOnlineTrendControl.1
WinCC push button control	PBUTTON.PbuttonCtrl.1
WinCC slider control	SLIDER.SliderCtrl.1
WinCC RulerControl	CCAxTrendRulerControl.AxRulerControl.1
WinCC UserArchiveControl	CCAxUserArchiveControl.AxUserArchiveControl.1

Example

In the following example, the ActiveX Control "WinCC Gauge Control" is inserted in the active picture.

```
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
```

See also

ActiveX controls (Page 1617)

HMIObjects Object (Listing) (Page 1884)

ActiveXControl Object (Page 1811)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
DefaultName (String)	The name of the folder that is to be created.

Example:

In the following example the folder "My Folder" will be created in the "Project Library":

```
Sub AddNewFolderToProjectLibrary()
'VBA120
Dim objProjectLib As HMISymbolLibrary
Set objProjectLib = Application.SymbolLibraries(2)
objProjectLib.FolderItems.AddFolder ("My Folder")
End Sub
```

See also

SymbolLibrary Object (Page 1962)

FolderItems Object (Listing) (Page 1868)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

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

Expression.AddFromClipboard(DefaultName)

Expression

Necessary. An expression or element which returns an object of the "FolderItems" type.

Parameters

Parameter (Data Type)	Description
DefaultName (String)	The name to be given to the object pasted into the components library.

Example:

In the following example the object "PC" from the "Global Library" will be copied into the folder "Folder 3" in the "Project Library":

```
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 1868)

SymbolLibrary Object (Page 1962)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

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

Parameter (Data Type)	Description
ObjectName (String)	The name of the object. You can address the object by its name in a listing.
ProgID (String)	The object type that is to be inserted. "Obtain the "ProgID" by prefixing the VBA object name with "HMI" "(e.g. HMICircle or HMIRectangle)

Example:

In the following example a circle will be inserted into the active picture and its background color set to "Red":

```
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 1922)
TextList Object (Page 1964)
StatusDisplay Object (Page 1959)
StaticText Object (Page 1956)
Slider object (Page 1952)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
PolyLine Object (Page 1928)
PictureWindow Object (Page 1919)
OptionGroup Object (Page 1916)
HMIObjects Object (Listing) (Page 1884)
Line Object (Page 1897)
IOField Object (Page 1886)
GraphicObject Object (Page 1870)
EllipseArc Object (Page 1856)
EllipseSegment Object (Page 1859)
Ellipse Object (Page 1853)
CircularArc Object (Page 1831)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)
BarGraph Object (Page 1819)
ApplicationWindow Object (Page 1817)
AddOLEObject Method (Page 1722)
AddActiveXControl Method (Page 1714)
VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
DefaultName (String)	The name to be given to the object pasted into the components library.
pHMIObject (HMIObject)	The object that is to be inserted into the Components Library from the specified picture.

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

```
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 1868)

SymbolLibrary Object (Page 1962)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

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.

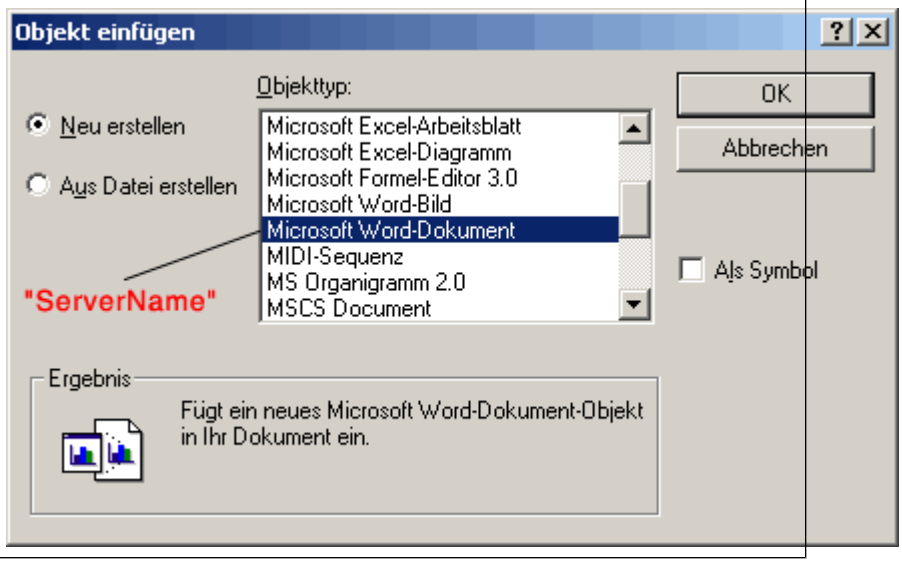
syntax

```
Expression.AddOLEObject(ObjectName, ServerName, [CreationType],
[UseSymbol])
```

Expression

Necessary. An expression or element which returns an object of the "HMIObjects" type.

Parameters

Parameter (Data Type)	Description
ObjectName (String)	The name of the object. You can address the object by its name in a listing.
ServerName (String)	The name of the application which is to contain the OLE Element, or the file name complete with path. The value for "ServerName" corresponds to the "Object Type" in the "Insert Object" dialog: 

Parameter (Data Type)	Description
CreationType (HMIOLEObjectCreation Type-)	Defines whether the OLE Element will be newly created or an existing file will be used: • HMIOLEObjectCreationTypeDirect: Corresponds to setting "Create New". This setting is used if you do not specify the parameter. • HMIOLEObjectCreationTypeByLink: Corresponds to setting "Create from File". 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 "ServerName" parameter. • HMIOLEObjectCreationTypeByLinkWithReference: Same as above, except that changes in OLE Element affect the original file. Assign a name to the file via the "ServerName" parameter.
UseSymbol (Boolean)	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.

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
```

See also

OLEObject Object (Page 1914)

HMIObjects Object (Listing) (Page 1884)

VBA Reference (Page 1664)

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:

```
Sub AlignSelectedObjectsBottom()
```

```
'VBA125
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.AlignBottom
End Sub
```

See also

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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 1814)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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 HMIObjects 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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

CascadeWindows Method

Description

Arranges all open pictures in the Graphics Designer in a cascade (i.e. overlapping).

syntax

Expression.`Methode(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:

```
Sub CascadeWindows()
'VBA130
Application.CascadeWindows
End Sub
```

See also

VBA Reference (Page 1664)

Application Object (Page 1814)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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:

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
CheckOK (Boolean)	TRUE if the syntax of the specified C script is correct.
Error (String)	The message text that is output if the C script is incorrect.

Example:

--

See also

DynamicDialog Object (Page 1851)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
FileName (String)	The name of the PDL file to be closed.

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 1847)

ActiveDocument Property (Page 1996)

Documents Object (Listing) (Page 1850)

VBA Reference (Page 1664)

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:

```
Sub CloseAllDocuments()
'VBA136
Application.Documents.CloseAll
End Sub
```

See also

Documents Object (Listing) (Page 1850)

VBA Reference (Page 1664)

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
MsgBox "The dynamic-dialog will be changed into a C-script."
.ConvertToScript
End With
End Sub
```

See also

DynamicDialog Object (Page 1851)

VBA Reference (Page 1664)

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
```

See also

Document Object (Page 1847)

ActiveDocument Property (Page 1996)

Selection Object (Listing) (Page 1949)

PasteClipboard Method (Page 1781)

Add Method (Documents Listing) (Page 1709)

Activate Method (Page 1704)

VBA Reference (Page 1664)

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":

```
Sub CopyObjectFromGlobalLibraryToProjectLibrary()
'VBA139
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")).Folder.AddFromClipBoard ("Copy of PC/PLC")
End Sub
```

See also

SymbolLibrary Object (Page 1962)

FolderItem Object (Page 1867)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

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:

```
Sub CreateCustomizedObject()
'VBA140
Dim objCircle As HMICircle
Dim objRectangle As HMIRectangle
Dim objCustObject As HMICustomizedObject
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

Selection Object (Listing) (Page 1949)

CustomizedObject Object (Page 1839)

VBA Reference (Page 1664)

Customized Objects (Page 1626)

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.

Parameter (Data Type)	Description
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 1932)

DeleteDynamic Method (Page 1747)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

Group Object (Page 1873)

VBA Reference (Page 1664)

Group Objects (Page 1618)

4.4.6.2 D-M

GetDeclutterObjectSize method

Description

Reads the limits displaying and hiding objects (decluttering) in the specified picture.

Syntax

Expression.`GetDeclutterObjectSize(Min, Max)`

Expression

Required. An expression or element which returns an object of the "Document" type.

Parameter

Parameter (data type)	Description
Min (Long)	Lower size range in pixels.
Max (Long)	Upper size range in pixels.

Example

In the following example, the decluttering limits of the active picture are read and output:

```
Sub ReadSettingsOfPicture()
'VBA848
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

- ToolbarItem
- VariableTrigger
- View

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 1892)

View Object (Page 1991)

VariableTrigger Object (Page 1989)

ToolbarItem Object (Page 1970)

FolderItem Object (Page 1867)

HMIObject Object (Page 1882)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

VBA Reference (Page 1664)

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:

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

DeleteDynamic Method

Description

Removes the dynamic characteristic from the specified property.

syntax

Expression.`DeleteDynamic`

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 1932)

CreateDynamic Method (Page 1742)

VBA Reference (Page 1664)

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:

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

Destroy Method

Description

Destroys the specified customized object. The objects remain intact.

syntax

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 1839)

Destroy Method (Page 1749)

Delete Method (Page 1745)

CreateCustomizedObject Method (Page 1741)

How to Edit a Customized Object with VBA (Page 1628)

DuplicateSelection Method

Description

Duplicates the objects selected in the specified picture. The objects created in this way are added to the HMIObjects 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.`DuplicateSelection()`

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

Selection Object (Listing) (Page 1949)

HMIObjects Object (Listing) (Page 1884)

VBA Reference (Page 1664)

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()`

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 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 1664)

Selection Object (Listing) (Page 1949)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

Export Method

Description

Saves the specified picture as an EMF file.

Syntax

Expression.`Export(Type, Path)`

Expression

Required. An expression or element which returns an object of the "Document" type.

Parameter

Parameter (Data Type)	Description
Type (HMIImportExportType)	Defines the format in which the exported picture will be saved.
Path (String)	The path in which the picture is going to be exported. The path must exist.

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 1991)

Document Object (Page 1847)

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.

Parameter (Data Type)	Description
ObjectType (String)	The object type that is to be searched for. Specify the "ProgID" of the object concerned. "Obtain the "ProgID" by prefixing the VBA object name with "HMI" "(e.g. HMICircle or HMIRectangle)
ObjectName (String)	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.
PropertyName (String)	The name of the object property that is to be searched for. Specify the VBA property name concerned (e.g. "BackColor" in place of "Background Color").

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:

```
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 2315)

Name Property (Page 2228)

Property Object (Page 1932)

HMIObjects Object (Listing) (Page 1884)

How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
DisplayName (String)	The display name of the object that is to be searched for in the Components Library.

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 1867)

Accessing the component library with VBA (Page 1593)

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:

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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:

```
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
ActiveDocument.Selection.FlipVertically
iPropertyValue = objStaticText.AlignmentLeft
MsgBox "Value of '" & strPropertyName & "' after flip: " & iPropertyValue
End With
End Sub
```

See also

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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:

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
PathName (String)	The internal access path on which the object is located in the Components Library.

Example:

In this example one object from the entire library will be returned and its display name will be output:

```
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 1962)

FolderItem Object (Page 1867)

Accessing the component library with VBA (Page 1593)

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.`InsertFromMenuItem(Position, Key, pMenuItem, DefaultToolTipText)`

Expression

Necessary. An expression or element which returns an object of the "ToolbarItems" type.

Parameters

Parameter (Data Type)	Description
Position (Long)	Defines the position of the icon within the user-defined toolbar.
Key (Variant)	Identifies the symbol. Use unique names for "Key" "(e.g. tItem1_1).
pMenuItem (HMIMenuItem)	The MenuItem object that is intended to be referenced.
DefaultToolTipText (String)	Defines for the icon concerned the tool tip text that will be displayed when you move the mouse over the icon.

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

ToolbarItems Object (Listing) (Page 1973)

InsertSeparator Method (Page 1766)

Add Method (CustomToolbars Listing) (Page 1708)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

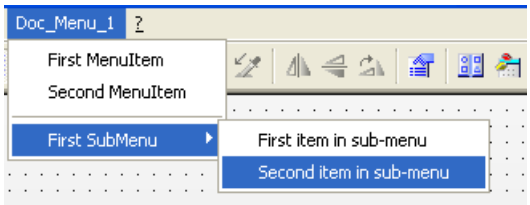
Parameter (Data Type)	Description
Position (Long)	Defines the position of the user-defined menu within the menu bar. However, picture-specific menus are always positioned to the right of application-specific menus.
Key (Variant)	Identifies the user-defined menu. Use unique names for "Key" (e.g. "DocMenu1")
DefaultLabel (String)	The name of the user-defined menu.

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 custumized 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 1904)

InsertSubmenu Method (Page 1767)

InsertSeparator Method (Page 1766)

InsertMenuItem Method (Page 1765)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

Parameter (Data Type)	Description
Position (Long)	Defines the position of the submenu within the user-defined menu.
Key (Variant)	Identifies the submenu. Use unique names for "Key" "(e.g. dSubMenu1_4).
DefaultLabel (String)	Defines the name of the submenu.

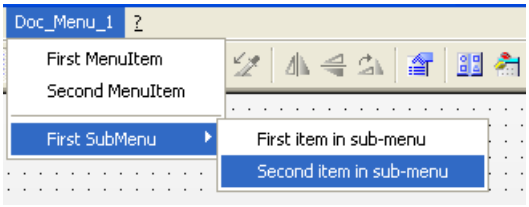
Example:

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")
'
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 menu structure.



See also

MenuItems Object (Listing) (Page 1909)

MenuItem Object (Page 1906)

InsertSubmenu Method (Page 1767)

InsertSeparator Method (Page 1766)

InsertMenu Method (Page 1763)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

Parameter (Data Type)	Description
Position (Long)	Defines the position of the dividing line within the user-defined menu or user-defined toolbar.
Key (Variant)	Identifies the dividing line. Use unique names for "Key" (e.g. "tSeparator1_2").

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 1973)

MenuItems Object (Listing) (Page 1909)

InsertToolbarItem Method (Page 1769)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

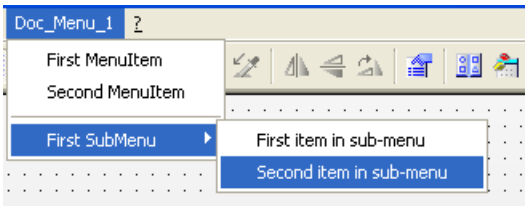
Parameter (Data Type)	Description
Position (Long)	Defines the position of the submenu within the user-defined menu.
Key (Variant)	Identifies the submenu. Use unique names for "Key" "(e.g. dSubMenu1_4).
DefaultLabel (String)	Defines the name of the submenu.

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()
'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")
'
'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 menu structure:



See also

MenuItem Object (Page 1906)

InsertSeparator Method (Page 1766)

InsertMenuItem Method (Page 1765)

InsertMenu Method (Page 1763)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

InsertToolbarItem Method

Description

Inserts a new icon in an existing user-defined toolbar.

syntax

Expression.`InsertToolbarItem(Position, Key, DefaultToolTipText)`

Expression

Necessary. An expression or element which returns an object of the "ToolbarItems" type.

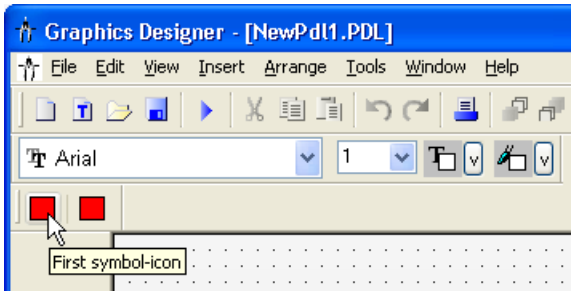
Parameters

Parameter (Data Type)	Description
Position (Long)	Defines the position of the icon within the user-defined toolbar.
Key (Variant)	Identifies the symbol. Use unique names for "Key" "(e.g. tItem1_1).
DefaultToolTipText (String)	Defines for the icon concerned the tool tip text that will be displayed when you move the mouse over the icon.

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 1973)

InsertSeparator Method (Page 1766)

Add Method (CustomToolbars Listing) (Page 1708)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

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

Parameter (Data Type)	Description
Index (Variant)	Defines the CS layer. Value range from 1 to 32. Layer0 corresponds to the index value "1".

Example:

The following example determines whether CS layer 1 in the copy of the active picture is visible and outputs the result:

```
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 1847)

VBA Reference (Page 1664)

Editing Layers with VBA (Page 1603)

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

Parameter (Data Type)	Description
Index (Variant)	Defines the RT layer. Value range from 1 to 32. Layer0 corresponds to the index value "1".

Example:

The following example determines whether RT layer 1 is visible and outputs the result:

```
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 1847)
VBA Reference (Page 1664)
Editing Layers with VBA (Page 1603)

Item Method

Description

Returns an element from a listing.

syntax

Expression.`Item(Index)`

Expression

Necessary. An expression or element which returns an object.

Parameters

Parameter (Data Type)	Description
Index (Variant)	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.

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 1987)
Count Property (Page 2082)
Views Object (Listing) (Page 1993)
VariableTriggers Object (Listing) (Page 1990)
ToolbarItems Object (Listing) (Page 1973)
Toolbars Object (Listing) (Page 1968)
SymbolLibraries Object (Listing) (Page 1963)
Selection Object (Listing) (Page 1949)
Properties Object (Listing) (Page 1931)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
MenuItems Object (Listing) (Page 1909)
Menus Object (Listing) (Page 1904)
Layers Object (Listing) (Page 1895)
LanguageTexts Object (Listing) (Page 1893)
LanguageFonts Object (Listing) (Page 1890)
GroupedObjects Object (Listing) (Page 1877)
FolderItems Object (Listing) (Page 1868)
Events Object (Listing) (Page 1863)
Documents Object (Listing) (Page 1850)
DataLanguages Object (Listing) (Page 1842)
ConnectionPoints Object (Listing) (Page 1837)
AnalogResultInfos Object (Listing) (Page 1813)
Actions Object (Listing) (Page 1810)
VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
LangID (Long)	This is the language identifier. The list of language identifiers is contained, for example, in the "Languages.csv" file that is found in the index of the WinCC documentation.

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):

```
Sub ExampleForLanguageFonts()
'VBA168
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 English:
With objLangFonts.ItemByLCID(1033)
.Family = "Times New Roman"
.Bold = False
.Italic = True
.Underlined = False
.Size = 14
End With
'
'To make fontsettings 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 1890)

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

Parameter (Data Type)	Description
FileName (String)	The name of the PDD file which it is intended to load.

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:

```
Sub LoadDefaultConfig()
'VBA169
Application.LoadDefaultConfig ("Test.PDD")
End Sub
```

See also

Application Object (Page 1814)

SaveDefaultConfig Method (Page 1793)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Necessary. An expression or element which returns an object of the "Document" or "Selection" type.

Parameters

Parameter (Data Type)	Description
PosX (Long)	The number of pixels by which the selection is to be moved horizontally.
PosY (Long)	The number of pixels by which the selection is to be moved vertically.

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
'Instead of next you can write "ActiveDocument.Selection.MoveSelection".
ActiveDocument.MoveSelection nPosX, nPosY
End Sub
```

See also

Top Property (Page 2311)

Left Property (Page 2192)

Document Object (Page 1847)

VBA Reference (Page 1664)

4.4.6.3 O-Z

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

Parameter (Data Type)	Description
FileName (String)	The name of the PDL file to be opened. Unless you saved the PDL file in the "GraCS" folder of the open project, you must also specify the path at the same time.
HMIOpenDocumentType (HMIDocumentType)	Defines how the picture will be opened: • HMIDocumentTypeVisible: Opens the picture for direct processing. This is the default setting if you do not specify the parameter. • 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.

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
```

See also

Hide Property (Page 2140)

Documents Object (Listing) (Page 1850)

VBA Reference (Page 1664)

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:

```
Sub CopySelectionToNewDocument()
'VBA176
Dim iNewDoc As String
ActiveDocument.CopySelection
Application.Documents.Add hmiOpenDocumentTypeVisible
iNewDoc = Application.Documents.Count
Application.Documents(iNewDoc).PasteClipboard
End Sub
```

See also

ActiveDocument Property (Page 1996)

Document Object (Page 1847)

CopySelection Method (Page 1738)

Add Method (Documents Listing) (Page 1709)

Activate Method (Page 1704)

VBA Reference (Page 1664)

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 1991)

VBA Reference (Page 1664)

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 1847)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
Index (Variant)	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.

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

Selection Object (Listing) (Page 1949)

GroupedObjects Object (Listing) (Page 1877)

VBA Reference (Page 1664)

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 1664)

Selection Object (Listing) (Page 1949)

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:

```
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 2139)

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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:

```
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 2408)

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

SameWidthAndHeight Method

Description

Sets the "Height" and "Width" properties for all selected objects in the specified picture to the smallest available value.

syntax

Expression.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:

```
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
```

See also

Width Property (Page 2408)

Height Property (Page 2139)

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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 1996)

Document Object (Page 1847)

VBA Reference (Page 1664)

SaveAll Method

Description

Saves all the open pictures in the Graphics Designer under their current names.

syntax

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:

```
Sub SaveAllDocuments()
'VBA185
Application.Documents.SaveAll
End Sub
```

See also

Documents Object (Listing) (Page 1850)

VBA Reference (Page 1664)

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

Expression.`SaveAs(FileName)`

Expression

Necessary. An expression or element which returns an object of the "Document" type.

Parameters

Parameter (Data Type)	Description
FileName (String)	The file name under which the picture is to be saved.

Example:

In the following example the active picture will be saved under the name "Test2.PDL":

```
Sub SaveDocumentAs()
'VBA186
ActiveDocument.SaveAs ("Test2.PDL")
End Sub
```

See also

LockedByCreatorID Property (Page 2200)

ActiveDocument Property (Page 1996)

Document Object (Page 1847)

VBA Reference (Page 1664)

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

Expression.SaveDefaultConfig(FileName)

Expression

Necessary. An expression or element which returns an object of the "Application" type.

Parameters

Parameter (Data Type)	Description
FileName (String)	The name of the PDD file.

Example:

In the following example the default settings for objects are saved to the file "Test.PDD".

```
Sub SaveDefaultConfig()
'VBA187
Application.SaveDefaultConfig ("Test.PDD")
End Sub
```

See also

Application Object (Page 1814)

LoadDefaultConfig Method (Page 1776)

VBA Reference (Page 1664)

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.

```
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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Selection Object (Listing) (Page 1949)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
Index (Variant)	Defines the CS layer that is going to be shown or hidden. Value range from 1 up to 32.
Val (Boolean)	TRUE if the specified CS layer is intended to be visible.

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 1847)

VBA Reference (Page 1664)

Editing Layers with VBA (Page 1603)

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

Parameter (Data Type)	Description
Min (Long)	Lower size range in pixels.
Max (Long)	Upper size range in pixels.

Example:

In the following example the settings for the lowest layer are configured in the active picture:

```
Sub ConfigureSettingsOfLayer()
'VBA190
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 decluttering of objects:
With ActiveDocument
.LayerDecluttering = True
```

```
.ObjectSizeDecluttering = True
.SetDeclutterObjectSize 50, 100
End With
End Sub
```

See also

ObjectSizeDecluttering Property (Page 2234)

Document Object (Page 1847)

VBA Reference (Page 1664)

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

Parameter (Data Type)	Description
Index (Variant)	Defines the RT layer that is going to be shown or hidden. Value range from 1 to 32.
Val (Boolean)	TRUE if the specified RT layer is intended to be visible.

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 1847)

VBA Reference (Page 1664)

Editing Layers with VBA (Page 1603)

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 1814)

VBA Reference (Page 1664)

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 1814)

VBA Reference (Page 1664)

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 1814)

VBA Reference (Page 1664)

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 1814)

VBA Reference (Page 1664)

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:

```
Sub TileWindowsVertically()
'VBA196
Application.TileWindowsVertically
End Sub
```

See also

Application Object (Page 1814)

VBA Reference (Page 1664)

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 1873)

CreateGroup Method (Page 1743)

VBA Reference (Page 1664)

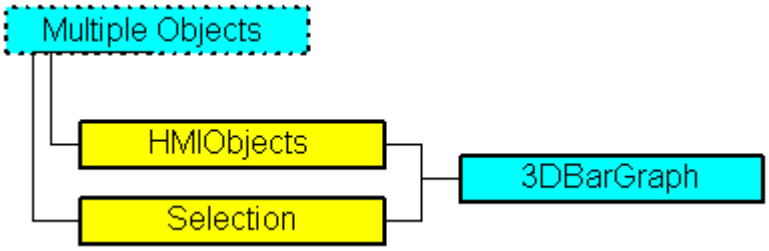
Group Objects (Page 1618)

4.4.7 Objects and Lists

4.4.7.1 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

Usage

Use the Add method to create a new "3D Bar" object in a picture:

```
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:

```
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
- Axe
- BorderColor
- Background
- BarDepth
- BarHeight
- BarWidth
- BaseX
- BaseY
- BorderColor
- BorderStyle
- BorderWidth
- Direction
- Height

- Layer00Checked
- Layer00Color
- Layer00Value
- Layer01Checked
- Layer01Color
- Layer01Value
- Layer02Checked
- Layer02Color
- Layer02Value
- Layer03Checked
- Layer03Color
- Layer03Value
- Layer04Checked
- Layer04Color
- Layer04Value
- Layer05Checked
- Layer05Color
- Layer05Value
- Layer06Checked
- Layer06Color
- Layer06Value
- Layer07Checked
- Layer07Color
- Layer07Value
- Layer08Checked
- Layer08Color
- Layer08Value
- Layer09Checked
- Layer09Color
- Layer09Value
- Layer10Checked
- Layer10Color
- Layer10Value
- Layer
- Left

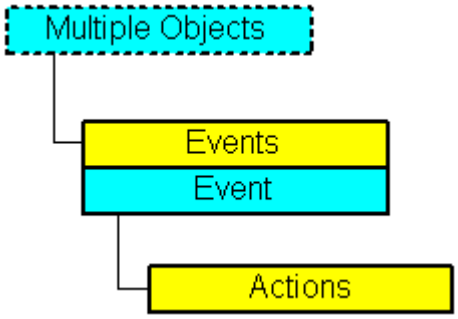
- LightEffect
- Max.
- Min.
- Name
- Operation
- PasswordLevel
- PredefinedAngles
- Process
- ToolTipText
- Top
- Visible
- Width
- ZeroPointValue

See also

Selection Object (Listing) (Page 1949)
Layer09Color Property (Page 2180)
Layer01Checked Property (Page 2163)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
ZeroPointValue Property (Page 2413)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
Process Property (Page 2260)
PredefinedAngels Property (Page 2259)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Min Property (Page 2225)
Max Property (Page 2207)
LightEffect Property (Page 2193)
Left Property (Page 2192)
Layer Property (Page 2160)
Layer10Value Property (Page 2183)
Layer10Color Property (Page 2182)
Layer10Checked Property (Page 2182)
Layer09Value Property (Page 2181)
Layer09Checked Property (Page 2180)
Layer08Value Property (Page 2179)
Layer08Color Property (Page 2178)
Layer08Checked Property (Page 2178)
Layer07Value Property (Page 2177)
Layer07Color Property (Page 2176)
Layer07Checked Property (Page 2176)
Layer06Value Property (Page 2175)
Layer06Color Property (Page 2174)
Layer06Checked Property (Page 2174)
Layer05Value Property (Page 2173)
Layer05Color Property (Page 2172)
Layer05Checked Property (Page 2172)

Actions Object (Listing)

Description



Displays a listing of the actions that are configured on an event.

VBA Object Name

HMIActions

Usage

Use the AddAction method to configure one or more actions on an event. 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:

```
Sub CreateVBActionToClickedEvent()
'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 1713)

Configuring Event-Driven Actions with VBA (Page 1644)

Parent Property (Page 2241)

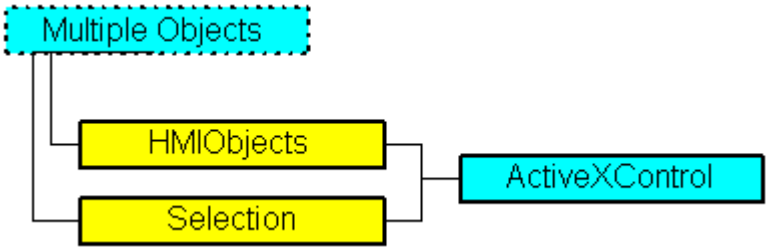
Item Property (Page 2152)

Count Property (Page 2082)

Application Property (Page 2008)

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.

```
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 2287)

AddActiveXControl Method (Page 1714)

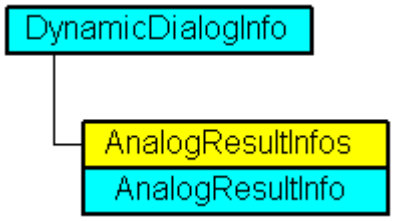
VBA Reference (Page 1664)

ActiveX controls (Page 1617)

ProgID Property (Page 2262)

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 1813)

Delete Method (Page 1745)

Value Property (Page 2331)

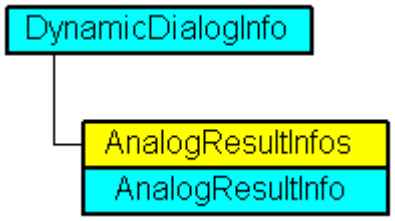
RangeTo Property (Page 2270)

Parent Property (Page 2241)

Application Property (Page 2008)

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:

```
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:

```
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 1706)

Parent Property (Page 2241)

Item Property (Page 2152)

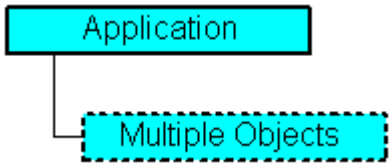
ElseCase Property (Page 2097)

Count Property (Page 2082)

Application Property (Page 2008)

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:

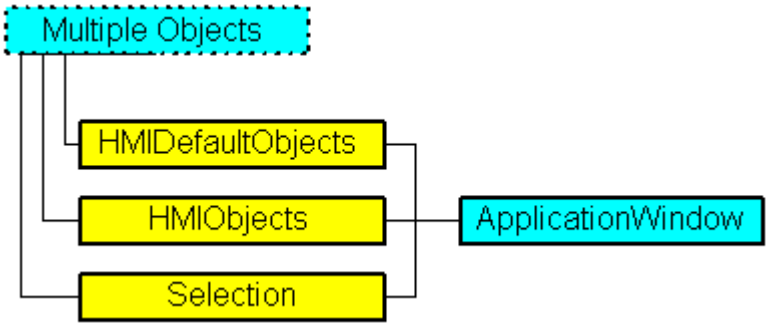
```
Sub ShowApplicationVersion()
'VBA208
MsgBox Application.Version
End Sub
```

See also

ShowTagDialog Method (Page 1801)
CurrentDesktopLanguage Property (Page 2084)
TileWindowsVertically Method (Page 1803)
TileWindowsHorizontally Method (Page 1802)
ShowSymbolLibraryDialog Method (Page 1800)
ShowPropertiesDialog Method (Page 1800)
SaveDefaultConfig Method (Page 1793)
LoadDefaultConfig Method (Page 1776)
CascadeWindows Method (Page 1731)
ArrangeMinimizedWindows Method (Page 1728)
Activate Method (Page 1704)
VBA Reference (Page 1664)
WindowState Property (Page 2412)
Visible Property (Page 2405)
Version Property (Page 2404)
VBE Property (Page 2404)
VBAVersion Property (Page 2404)
SymbolLibraries Property (Page 2299)
ProjectType Property (Page 2263)
ProjectName Property (Page 2262)
ProfileName Property (Page 2261)
Parent Property (Page 2241)
Name Property (Page 2228)
IsConnectedToProject Property (Page 2149)
Documents Property (Page 2095)
DefaultHMIObjects Property (Page 2089)
CustomToolbars Property (Page 2086)
CustomMenus Property (Page 2086)
CurrentDataLanguage Property (Page 2083)
ConfigurationFileName Property (Page 2081)
AvailableDataLanguages Property (Page 2013)
ApplicationDataPath Property (Page 2008)
Application Property (Page 2008)
ActiveDocument Property (Page 1996)

ApplicationWindow Object

Description



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:
```

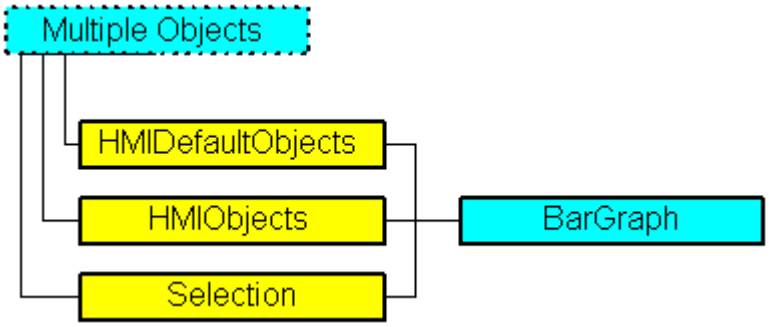
```
Sub ShowNameOfFirstSelectedObject()
'VBA211
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Caption Property (Page 2053)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
WindowBorder Property (Page 2409)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
Sizeable Property (Page 2292)
OnTop Property (Page 2236)
Name Property (Page 2228)
Moveable Property (Page 2227)
MaximizeButton Property (Page 2208)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
CloseButton Property (Page 2065)
Application Property (Page 2008)

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.

VBAbezeichnung

HMIBarGraph

Usage

Use the Add method to create a new "Bar" object in a picture:

```
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:

```
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:

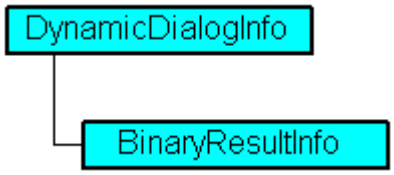
```
Sub ShowNameOfFirstSelectedObject()
'VBA214
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

TypeWarningHigh Property (Page 2322)
Max Property (Page 2207)
FillColor Property (Page 2104)
BorderStyle Property (Page 2043)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
ZeroPointValue Property (Page 2413)
ZeroPoint Property (Page 2412)
Width Property (Page 2408)
WarningLow Property (Page 2407)
WarningHigh Property (Page 2406)
Visible Property (Page 2405)
TypeWarningLow Property (Page 2323)
TypeToleranceLow Property (Page 2322)
TypeToleranceHigh Property (Page 2321)
TypeLimitLow5 Property (Page 2320)
TypeLimitLow4 Property (Page 2319)
TypeLimitHigh5 Property (Page 2318)
TypeLimitHigh4 Property (Page 2318)
TypeAlarmLow Property (Page 2317)
TypeAlarmHigh Property (Page 2316)
Trend Property (Page 2313)
TrendColor Property (Page 2314)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
ToleranceLow Property (Page 2308)
ToleranceHigh Property (Page 2308)
ScalingType Property (Page 2279)
Scaling Property (Page 2278)
ScaleTicks Property (Page 2277)
ScaleColor Property (Page 2277)
RightComma Property (Page 2273)
Process Property (Page 2260)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)

BinaryResultInfo Object

Description



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 1664)

PositiveValue Property (Page 2258)

Parent Property (Page 2241)

NegativeValue Property (Page 2229)

Application Property (Page 2008)

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")
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 1745)

VBA Reference (Page 1664)

BitSetValue Property (Page 2033)

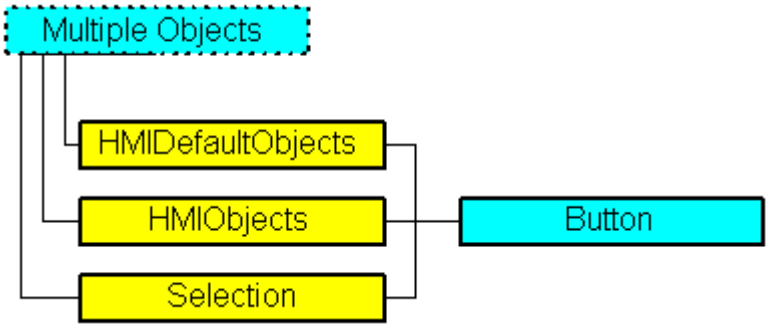
BitNumber Property (Page 2032)

BitNotSetValue Property (Page 2031)

Application Property (Page 2008)

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.

VBAbezeichnung

HMIButton

Usage

Use the Add method to create a new "Button" object in a picture:

```
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:

```
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:

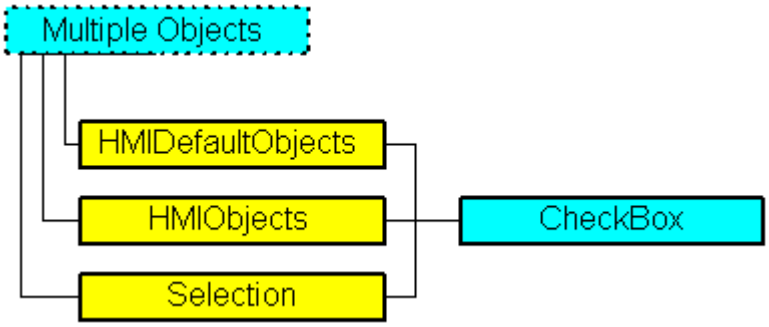
```
Sub ShowNameOfFirstSelectedObject()
'VBA219
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ForeFlashColorOn Property (Page 2133)
BorderColorBottom Property (Page 2038)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
Text Property (Page 2306)
PictureUp Property (Page 2253)
PictureDown Property (Page 2251)
PasswordLevel Property (Page 2244)
Orientation Property (Page 2239)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Hotkey Property (Page 2142)
Height Property (Page 2139)
ForeFlashColorOff Property (Page 2132)
ForeColor Property (Page 2132)
FontUnderline Property (Page 2131)
FontSize Property (Page 2130)
FontName Property (Page 2129)
FontItalic Property (Page 2128)
FontBold Property (Page 2126)
FlashRateForeColor Property (Page 2123)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashForeColor Property (Page 2114)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)

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.

VBAbezeichnung

HMICheckBox

Usage

Use the Add method to create a new "Check Box" object in a picture:

```
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:

```
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:

```
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

Selection Object (Listing) (Page 1949)

HMIObjects Object (Listing) (Page 1884)

HMIDefaultObjects Object (Listing) (Page 1878)

AddHMIObject Method (Page 1719)

VBA Reference (Page 1664)

Editing Objects with VBA (Page 1606)

BitSetValue Property (Page 2033)

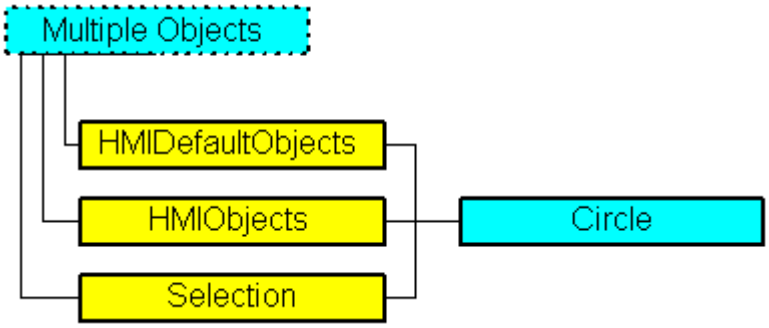
BitNumber Property (Page 2032)

BitNotSetValue Property (Page 2031)

Application Property (Page 2008)

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:

```
Sub AddCircle()
'VBA223
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("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:

```
Sub EditCircle()
'VBA224
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects("Circle")
objCircle.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

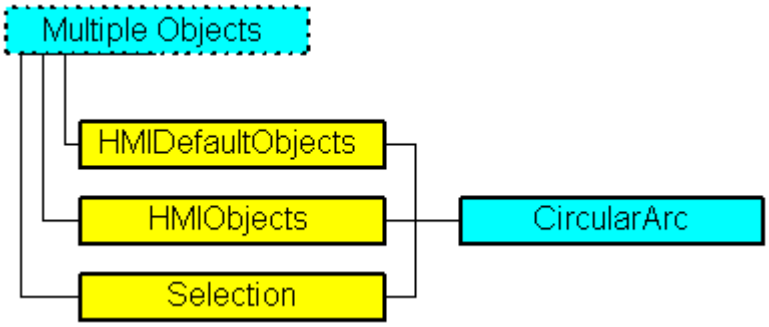
```
Sub ShowNameOfFirstSelectedObject()
'VBA225
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

FillColor Property (Page 2104)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
Radius Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

CircularArc Object

Description



Represents the "Circular Arc" object. The CircularArc 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

HMICircularArc

Usage

Use the Add method to create a new "Circular Arc" object in a picture:

```
Sub AddCiruclarArc()
'VBA226
Dim objCiruclarArc As HMICircularArc
Set objCiruclarArc = 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:

```
Sub EditCiruclarArc()
'VBA227
Dim objCiruclarArc As HMICircularArc
Set objCiruclarArc = ActiveDocument.HMIObjects("CircularArc")
objCiruclarArc.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

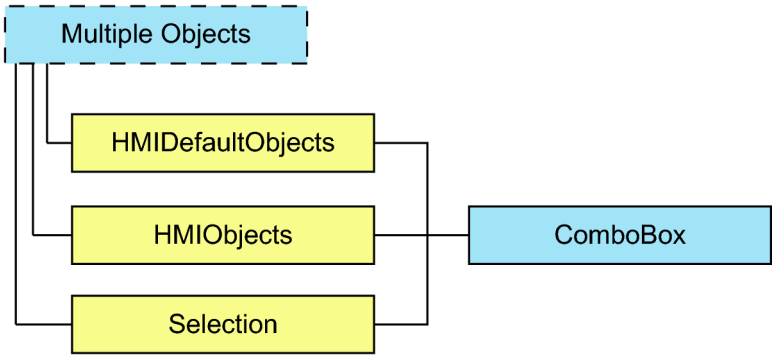
```
Sub ShowNameOfFirstSelectedObject()
'VBA228
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

HMIObjects Object (Listing) (Page 1884)
BorderBackColor Property (Page 2035)
Selection Object (Listing) (Page 1949)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
StartAngle Property (Page 2296)
Radius Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashBorderColor Property (Page 2112)
EndAngle Property (Page 2098)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)

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:

```
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:

```
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:

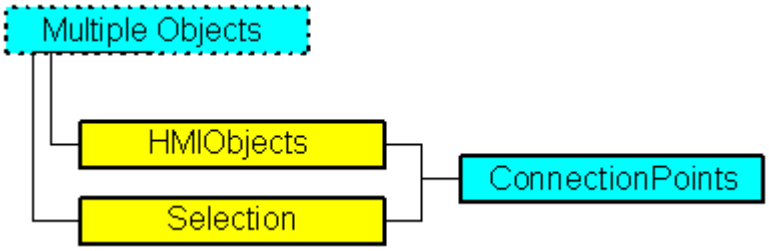
```
Sub ShowNameOfFirstSelectedObject()
'VBA824
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ObjectName Property (Page 2232)
Left Property (Page 2192)
Layer Property (Page 2160)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
NumberLines Property (Page 2231)
ForeColor Property (Page 2132)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackColor Property (Page 2017)
BorderStyle Property (Page 2043)
BorderWidth Property (Page 2045)
FillColor Property (Page 2104)
FillStyle Property (Page 2108)
FontName Property (Page 2129)
FontSize Property (Page 2130)
FontBold Property (Page 2126)
FontItalic Property (Page 2128)
FontUnderline Property (Page 2131)
AlignmentLeft Property (Page 2004)
GlobalShadow property (Page 2135)
Index Property (Page 2145)
Text Property (Page 2306)
Operation Property (Page 2237)
PasswordLevel Property (Page 2244)
Visible Property (Page 2405)
ToolTipText Property (Page 2310)
OperationMessage Property (Page 2238)
OperationReport Property (Page 2238)
SelText property (Page 2287)
SelIndex property (Page 2286)

ConnectionPoints Object (Listing)

Description



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:

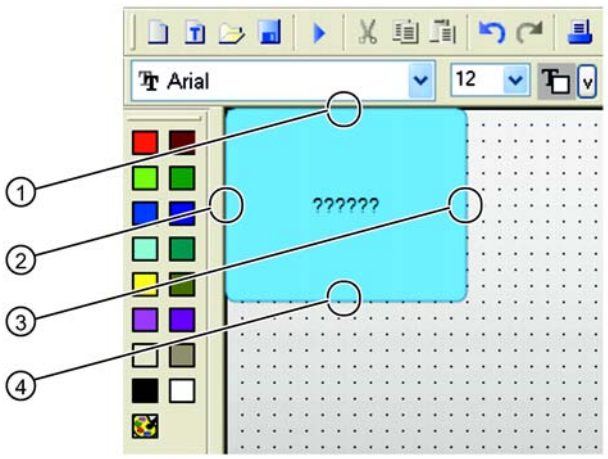
```
Sub CountConnectionPoints()
'VBA229
Dim objRectangle As HMIRectangle
Dim objConnPoints As HMIConnectionPoints
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
Set objConnPoints = ActiveDocument.HMIObjects("Rectangle1").ConnectionPoints
MsgBox "Rectangle1 has " & objConnPoints.Count & " connectionpoints."
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 2241)

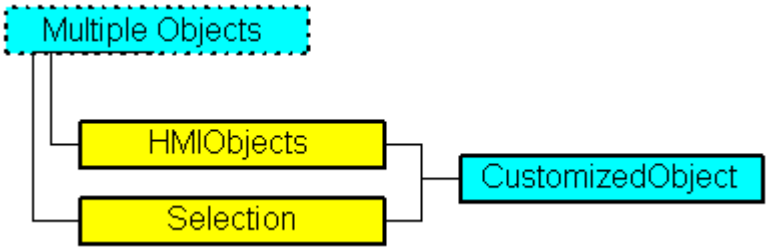
Item Property (Page 2152)

Count Property (Page 2082)

Application Property (Page 2008)

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

Usage

Use the CreateCustomizedObject Method with the Selection listing to create a new "Customized Object" object in a picture:

```
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

Selection Object (Listing) (Page 1949)

HMIObjects Object (Listing) (Page 1884)

Destroy Method (Page 1749)

Delete Method (Page 1745)

CreateCustomizedObject Method (Page 1741)

How to Edit a Customized Object with VBA (Page 1628)

VBA Reference (Page 1664)

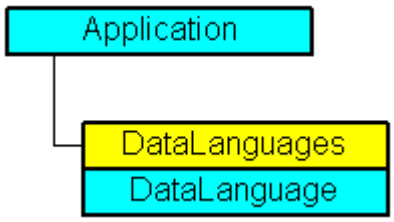
Customized Objects (Page 1626)

Editing Objects with VBA (Page 1606)

4.4.7.2 D-I

DataLanguage Object

Description



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 full list of language codes is available in the WinCC documentation (Index > Language ID). 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:

```
Sub ShowFirstObjectOfCollection()
'VBA232
```

```
Dim strName As String
strName = ActiveDocument.Application.AvailableDataLanguages(1).LanguageName
MsgBox strName
End Sub
```

See also

DataLanguages Object (Listing) (Page 1842)

VBA Reference (Page 1664)

Language-Dependent Configuration with VBA (Page 1570)

Parent Property (Page 2241)

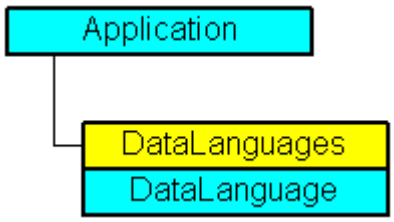
LanguageName Property (Page 2158)

LanguageID Property (Page 2158)

Application Property (Page 2008)

DataLanguages Object (Listing)

Description



A listing of the DataLanguage objects that represent all the installed project languages.

VBA Object Name

HMIDataLanguages

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 1570)

DataLanguage Object (Page 1841)

Item Method (Page 1772)

VBA Reference (Page 1664)

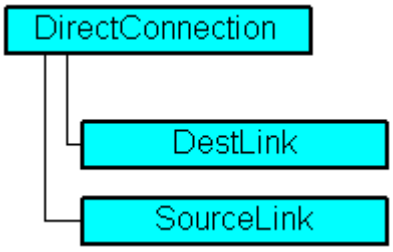
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

DestLink Object

Description



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 1845)

VBA Reference (Page 1664)

Type Property (Page 2315)

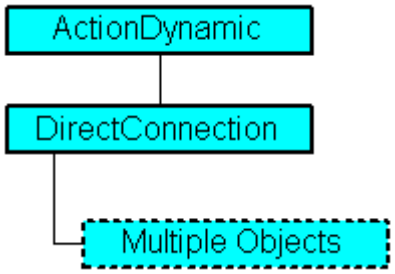
ObjectName Property (Page 2232)

AutomationName Property (Page 2011)

DestinationLink Property (Page 2090)

DirectConnection Object

Description



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:

```
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"
'
'Targetobject: Property "Left" of "Rectangle_B"
.DestinationLink.Type = hmiDestTypeProperty
.DestinationLink.ObjectName = "Rectangle_B"
.DestinationLink.AutomationName = "Left"
End With
End Sub
```

See also

DestinationLink Property (Page 2090)

SourceLink Object (Page 1955)

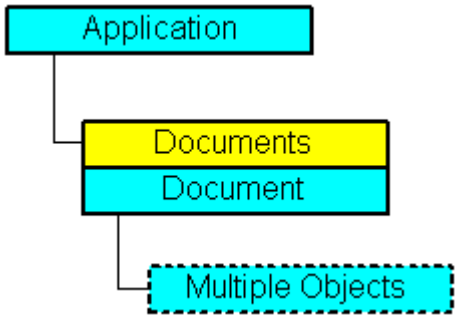
DestLink Object (Page 1843)

VBA Reference (Page 1664)

SourceLink Property (Page 2293)

Document Object

Description



Displays a picture in Graphics Designer. The document object is an element of the documents listing.

VBA Object Name

HMIDocument

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:

```
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":

```
Sub SaveDocumentAs()
```

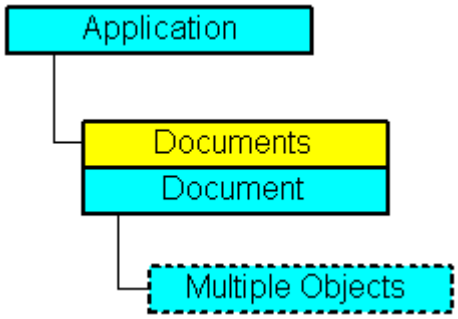
```
'VBA237
Application.Documents(3).SaveAs ("CopyOfPicture1")
End Sub
```

See also

Editing Pictures with VBA (Page 1601)
GridHeight Property (Page 2136)
Documents Object (Listing) (Page 1850)
SetRTLayerVisible Method (Page 1799)
SaveAs Method (Page 1792)
Save Method (Page 1791)
PrintProjectDocumentation Method (Page 1783)
PasteClipboard Method (Page 1781)
MoveSelection Method (Page 1779)
IsRTLayerVisible Method (Page 1771)
Export Method (Page 1754)
CopySelection Method (Page 1738)
Close Method (Page 1735)
VBA Reference (Page 1664)
ExtendedZoomingEnable Property (Page 2103)
Width Property (Page 2408)
Visible Property (Page 2405)
Views Property (Page 2405)
UpdateCycle Property (Page 2327)
TabOrderOtherAction Property (Page 2303)
TabOrderMouse Property (Page 2302)
TabOrderKeyboard Property (Page 2301)
TabOrderAllHMIObjects Property (Page 2300)
SnapToGrid Property (Page 2293)
HMIDefaultObjects Object (Listing) (Page 1878)
Selection Property (Page 2286)
Properties Property (Page 2264)
Path Property (Page 2244)
PasswordLevel Property (Page 2244)
Parent Property (Page 2241)
Operation Property (Page 2237)
Name Property (Page 2228)
LockedByCreatorID Property (Page 2200)
LastChange Property (Page 2160)
HMIObjects Property (Page 2141)
Hide Property (Page 2140)
Height Property (Page 2139)
GridWidth Property (Page 2137)
GridColor Property (Page 2136)
Grid Property (Page 2135)

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:

```
Sub ShowDocuments()
'VBA238
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:

```
Sub AddNewDocument()
'VBA239
Dim objDocument As Document
Set objDocument = Application.Documents.Add
End Sub
```

See also

Add Method (Page 1705)

Document Object (Page 1847)

SaveAll Method (Page 1792)

Open Method (Page 1780)

Item Method (Page 1772)

CloseAll Method (Page 1736)

Close Method (Page 1735)

VBA Reference (Page 1664)

Editing Pictures with VBA (Page 1601)

Parent Property (Page 2241)

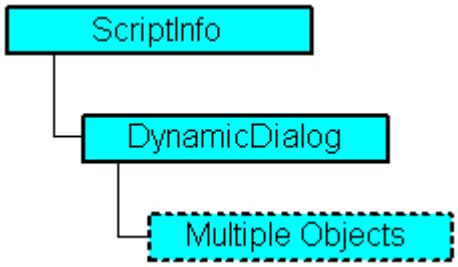
Count Property (Page 2082)

Application Property (Page 2008)

ActiveDocument Property (Page 1996)

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 1745)

ConvertToScript Method (Page 1737)

CheckSyntax Method (Page 1734)

VariableStateValues Property (Page 2401)

VariableStateChecked Property (Page 2400)

Trigger Property (Page 2314)

SourceCode Property (Page 2295)

ScriptType Property (Page 2280)

ResultType Property (Page 2272)

Parent Property (Page 2241)

Compiled Property (Page 2080)

BitResultInfo Property (Page 2033)

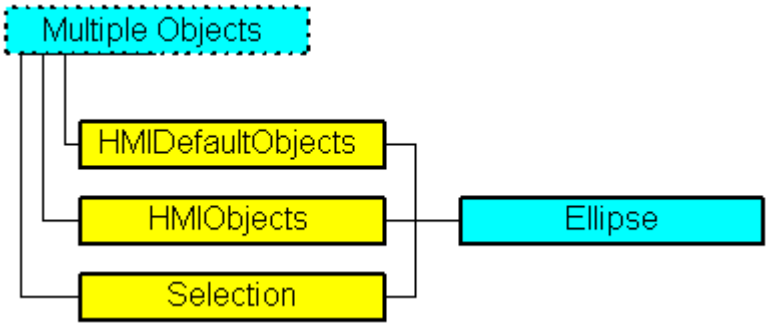
BinaryResultInfo Property (Page 2031)

Application Property (Page 2008)

AnalogResultInfos Property (Page 2006)

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:

```
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.

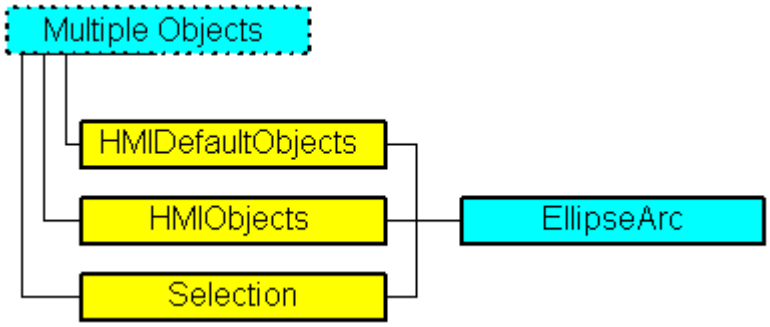
```
Sub ShowNameOfFirstSelectedObject()
'VBA243
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

FillingIndex Property (Page 2107)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
RadiusWidth Property (Page 2269)
RadiusHeight Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

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:

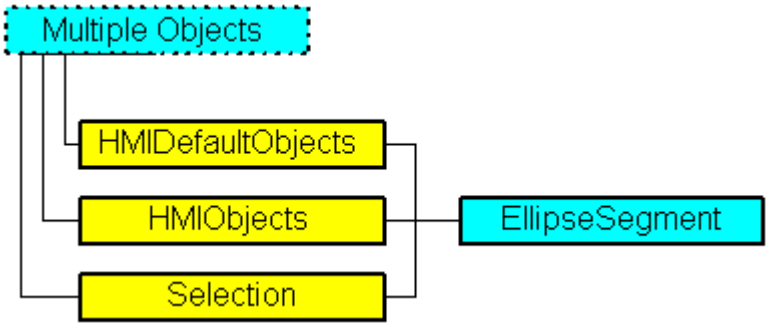
```
Sub EditDefaultPropertiesOfEllipseArc()
'VBA247
Dim objEllipseArc As HMIEllipseArc
Set objEllipseArc = Application.DefaultHMIObjects("HMIEllipseArc")
objEllipseArc.BorderColor = RGB(255, 255, 0)
'create new "EllipseArc"-object
Set objEllipseArc = ActiveDocument.HMIObjects.AddHMIObject("EllipseArc2", "HMIEllipseArc")
End Sub
```

See also

ToolTipText Property (Page 2310)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
StartAngle Property (Page 2296)
RadiusWidth Property (Page 2269)
RadiusHeight Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashBorderColor Property (Page 2112)
EndAngle Property (Page 2098)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)

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:

```
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:

```
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:

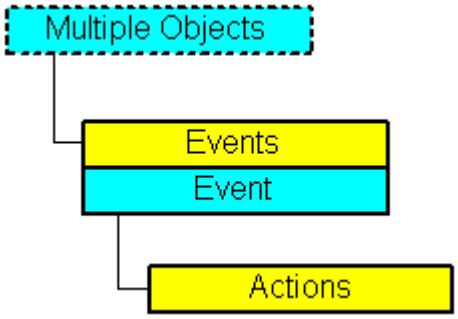
```
Sub ShowNameOfFirstSelectedObject()
'VBA250
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ToolTipText Property (Page 2310)
BackFlashColorOn Property (Page 2022)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
StartAngle Property (Page 2296)
RadiusWidth Property (Page 2269)
RadiusHeight Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
EndAngle Property (Page 2098)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

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 2008)

Delete Method (Page 1745)

AddAction Method (Page 1713)

VBA Reference (Page 1664)

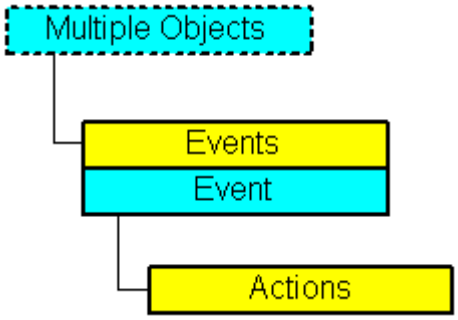
Parent Property (Page 2241)

EventType Property (Page 2101)

Actions Property (Page 1995)

Events Object (Listing)

Description



A listing of the Event objects that represent all the events configured onto an object. Use the Item method to define the event that is intended to be configured:

- 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):

Index	EventType (depending upon the object used)
0	hmiEventTypeNotDefined
1	hmiEventTypeMouseClick
2	hmiEventTypeMouseLButtonDown
3	hmiEventTypeMouseLButtonUp
4	hmiEventTypeMouseRButtonDown
5	hmiEventTypeMouseRButtonUp
6	hmiEventTypeKeyboardDown
7	hmiEventTypeKeyboardUp
8	hmiEventTypeFocusEnter
9	hmiEventTypeObjectChange
10	hmiEventTypePictureOpen

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.

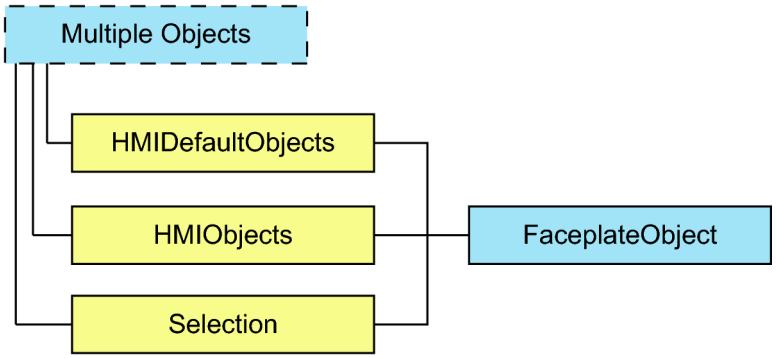
```
Sub ShowEventsOfAllObjectsInActiveDocument()
'VBA252
Dim colEvents As HMIEvents
Dim objEvent As HMIEvent
Dim iMax As Integer
Dim iIndex As Integer
Dim iAnswer As Integer
Dim strEventName As String
Dim strObjectName As String
Dim varEventType As Variant
iIndex = 1
iMax = ActiveDocument.HMIObjects.Count
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 1772)
VBA Reference (Page 1664)
Parent Property (Page 2241)
Count Property (Page 2082)
Application Property (Page 2008)

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:

```
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:

```
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:

```
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 2232)

Layer Property (Page 2160)

Left Property (Page 2192)

Top Property (Page 2311)

Width Property (Page 2408)

Height Property (Page 2139)

Operation Property (Page 2237)

PasswordLevel Property (Page 2244)

Visible Property (Page 2405)

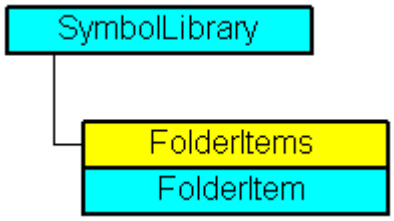
ToolTipText Property (Page 2310)

ScalingMode property (Page 2279)

FaceplateType property (Page 2104)

FolderItem Object

Description



Represents 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.

```
Sub CopyFolderItemToClipboard()
'VBA254
Dim objGlobalLib As HMISymbolLibrary
Set objGlobalLib = Application.SymbolLibraries(1)
objGlobalLib.FolderItems("Folder2").Folder("Folder2").Folder.Item("Object1").CopyToClipboa
rd
End Sub
```

See also

Type Property (Page 2315)

FolderItems Object (Listing) (Page 1868)

Delete Method (Page 1745)

CopyToClipboard Method (Page 1740)

How to Insert an Object from the Component Library into a Picture with VBA (Page 1599)

How to Edit the Component Library with VBA (Page 1596)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

Parent Property (Page 2241)

Name Property (Page 2228)

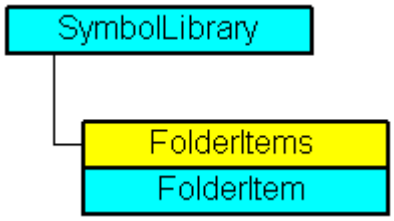
LDNames Property (Page 2187)

Folder Property (Page 2124)

Application Property (Page 2008)

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:

```
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":

```
Sub AddNewFolderToProjectLibrary()
'VBA256
Dim objProjectLib As HMISymbolLibrary
Set objProjectLib = Application.SymbolLibraries(2)
objProjectLib.FolderItems.AddFolder ("My Folder")
End Sub
```

See also

AddItem Method (Page 1720)

SymbolLibrary Object (Page 1962)

FolderItem Object (Page 1867)

Item Method (Page 1772)

AddFromClipboard Method (Page 1717)

AddFolder Method (Page 1716)

How to Insert an Object from the Component Library into a Picture with VBA (Page 1599)

How to Edit the Component Library with VBA (Page 1596)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

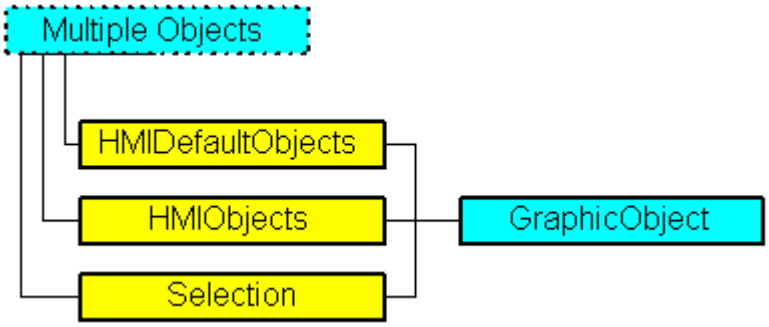
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

GraphicObject Object

Description



Represents the object called "Graphic Object". The GraphicObject 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

HMIGraphicObject

Usage

Use the Add method to create a new "Graphic Object" object in a picture:

```
Sub AddGraphicObject()
'VBA257
Dim objGraphicObject As HMIGraphicObject
Set objGraphicObject = ActiveDocument.HMIObjects.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:

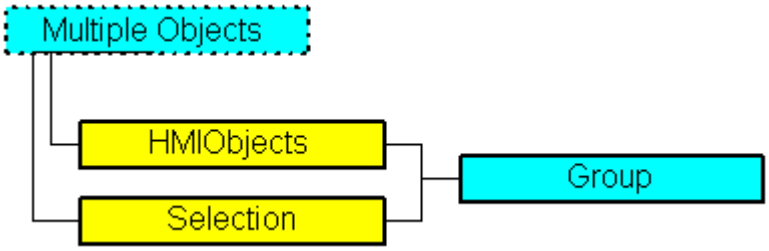
```
Sub ShowNameOfFirstSelectedObject()
'VBA259
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Left Property (Page 2192)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
PicUseTransColor Property (Page 2256)
PictureName Property (Page 2252)
PicTransColor Property (Page 2250)
PicReferenced Property (Page 2249)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

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:

```
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:

```
Sub EditGroup()
'VBA261
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("Group-Object")
MsgBox objGroup.ObjectName
End Sub
```

See also

Selection Object (Listing) (Page 1949)

HMIObjects Object (Listing) (Page 1884)

GroupedObjects Object (Listing) (Page 1877)

Ungroup Method (Page 1803)

Remove Method (Page 1784)

Delete Method (Page 1745)

Add Method (GroupedObjects Listing) (Page 1710)

How to Edit Objects in a Group Object with VBA (Page 1623)

How to Edit Group Objects with VBA (Page 1620)

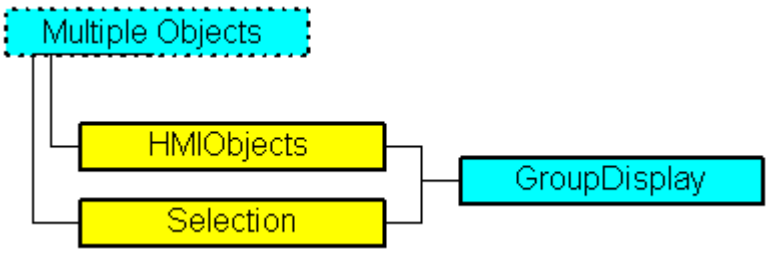
VBA Reference (Page 1664)

Group Objects (Page 1618)

Editing Objects with VBA (Page 1606)

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:

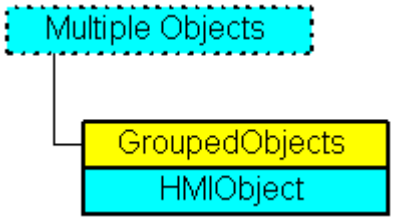
```
Sub ShowNameOfFirstSelectedObject()
'VBA264
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

MCText Property (Page 2222)
Height Property (Page 2139)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
UserValue4 Property (Page 2330)
UserValue3 Property (Page 2329)
UserValue2-Eigenschaft (Page 2328)
UserValue1 Property (Page 2328)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
SignificantMask Property (Page 2290)
SameSize Property (Page 2276)
Relevant Property (Page 2272)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
MessageClass Property (Page 2224)
MCKQTextFlash Property (Page 2221)
MCKQTextColorOn Property (Page 2220)
MCKQTextColorOff Property (Page 2220)
MCKQBackFlash Property (Page 2219)
MCKQBackColorOn Property (Page 2218)
MCKQBackColorOff Property (Page 2218)
MCKOTextFlash Property (Page 2217)
MCKOTextColorOn Property (Page 2216)
MCKOTextColorOff Property (Page 2215)
MCKOBackFlash Property (Page 2215)
MCKOBackColorOn Property (Page 2214)
MCKOBackColorOff Property (Page 2213)
MCGUTextFlash Property (Page 2213)
MCGUTextColorOn Property (Page 2212)
MCGUTextColorOff Property (Page 2211)
MCGUBackFlash Property (Page 2211)
MCGUBackColorOn Property (Page 2210)
MCGUBackColorOff-Eigenschaft (Page 2209)

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":

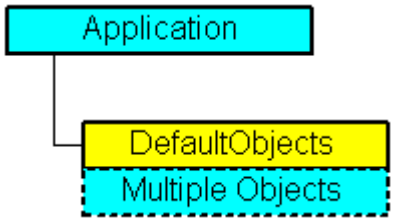
```
Sub RemoveObjectFromGroup()
'VBA266
Dim objGroup As HMIGroup
Set objGroup = ActiveDocument.HMIObjects("Group1")
objGroup.GroupedHMIObjects.Remove (1)
End Sub
```

See also

Item Method (Page 1772)
Group Object (Page 1873)
Remove Method (Page 1784)
Add Method (GroupedObjects Listing) (Page 1710)
How to Edit Group Objects with VBA (Page 1620)
VBA Reference (Page 1664)
Group Objects (Page 1618)
Parent Property (Page 2241)
GroupedHMIObjects Property (Page 2138)
Count Property (Page 2082)
Application Property (Page 2008)

HMIDefaultObjects Object (Listing)

Description



A listing of the following HMIObject objects:

Object	VBA object name
Line	HMILine
Polygon	HMIPolygon
Polyline	HMIPolyLine
Ellipse	HMIEllipse
Circle	HMICircle
Ellipse segment	HMIEllipseSegment
Pie segment	HMIPieSegment
Ellipse arc	HMIEllipseArc
Circular arc	HMICircularArc
Rectangle	HMIRectangle
Rounded rectangle	HMIRoundRectangle
Application window	HMIApplicationWindow
Screen Window	HMIPictureWindow
Static text	HMIStaticText

Object	VBA object name
I/O Field	HMIIOField
Button	HMIButton
Check box	HMICheckBox
Radio box	HMIOptionGroup
Round button	HMIRoundButton
Bar	HMIBarGraph
Slider object	HMISlider
Graphic Object	HMIGraphicObject
Status display	HMIStatusDisplay
Text list	HMITextList
Connector	HMIObjConnection
Multiple row text	HMIMultiLineEdit
Combo box	HMIComboBox
List box	HMIListBox
Polygon tube	HMITubePolyline
T-piece	HMITubeTeeObject
Double T-piece	HMITubeDoubleTeeObject
Tube bend	HMITubeArcObject
3D bar	HMI3DBarGraph
Group display	HMIGroupDisplay
Faceplate instance	HMIFaceplateObject

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:

```
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
```

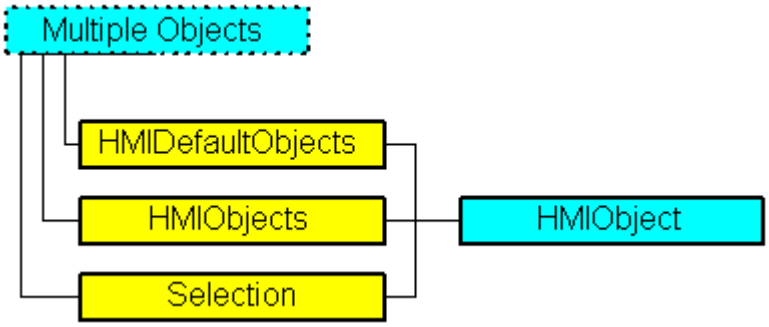
```
strMessage = strMessage & "Element: " & iIndex & " / Objecttype: " & strType & " /
Objectname: " & strName
End With
If 0 = iIndex Mod 10 Then
MsgBox strMessage
strMessage = ""
Else
strMessage = strMessage & vbCrLf & vbCrLf
End If
Next iIndex
MsgBox "Element: " & iIndex & vbCrLf & "Objecttype: " & strType & vbCrLf & "Objectname: "
& strName
End Sub
```

See also

Button Object (Page 1824)
TextList Object (Page 1964)
StatusDisplay Object (Page 1959)
StaticText Object (Page 1956)
Slider object (Page 1952)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
PolyLine Object (Page 1928)
Polygon Object (Page 1925)
PieSegment Object (Page 1922)
PictureWindow Object (Page 1919)
OptionGroup Object (Page 1916)
Line Object (Page 1897)
IOField Object (Page 1886)
EllipseSegment Object (Page 1859)
EllipseArc Object (Page 1856)
Ellipse Object (Page 1853)
CircularArc Object (Page 1831)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
BarGraph Object (Page 1819)
ApplicationWindow Object (Page 1817)
Item Method (Page 1772)
VBA Reference (Page 1664)
Parent Property (Page 2241)
Count Property (Page 2082)
DefaultHMIObjects Property (Page 2089)
Application Property (Page 2008)

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:

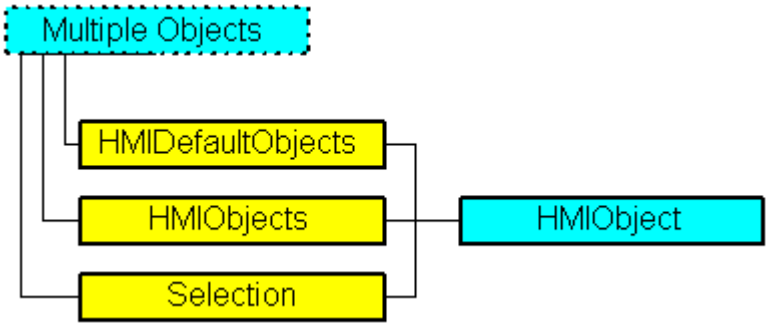
```
Sub DeleteObject()
'VBA269
ActiveDocument.HMIObjects(1).Delete
End Sub
```

See also

Name Property (Page 2228)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
Delete Method (Page 1745)
VBA Reference (Page 1664)
Default objects, Smart objects, Windows objects and Tube objects (Page 1609)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Type Property (Page 2315)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
TabOrderAlpha Property (Page 2301)
TabOrderSwitch Property (Page 2299)
Selected Property (Page 2285)
Properties Property (Page 2264)
PasswordLevel Property (Page 2244)
Parent Property (Page 2241)
Operation Property (Page 2237)
Left Property (Page 2192)
LDTooltipTexts Property (Page 2190)
Layer Property (Page 2160)
Height Property (Page 2139)
GroupParent Property (Page 2138)
Events Property (Page 2099)
Application Property (Page 2008)

HMIObjects Object (Listing)

Description



A listing of the HMIObject objects that represent all the objects in the picture.

VBA Object Name

HMIObjects

Note

Die Reihenfolge der HMI-Objekte in der HMIObjects-Auflistung kann sich durch Hinzufügen und/oder Löschen von HMI-Objekten ändern.

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:

```
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:

```
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:

```
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 2082)

HMIDefaultObjects Object (Listing) (Page 1878)

Selection Object (Listing) (Page 1949)

Item Method (Page 1772)

Find Method (Page 1755)

AddOLEObject Method (Page 1722)

AddHMIObject Method (Page 1719)

AddActiveXControl Method (Page 1714)

How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)

VBA Reference (Page 1664)

Default objects, Smart objects, Windows objects and Tube objects (Page 1609)

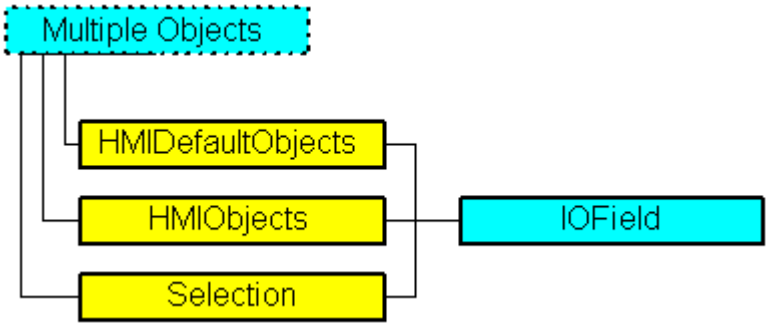
Editing Objects with VBA (Page 1606)

Parent Property (Page 2241)

Application Property (Page 2008)

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:

```
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
```

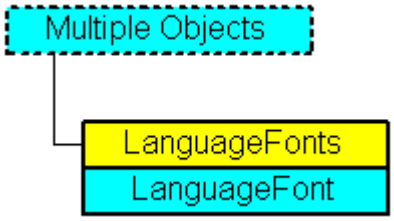
See also

LimitMin Property (Page 2197)
ClearOnNew Property (Page 2064)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
PasswordLevel Property (Page 2244)
OutputValue Property (Page 2241)
OutputFormat Property (Page 2240)
Orientation Property (Page 2239)
OperationReport Property (Page 2238)
OperationMessage Property (Page 2238)
Operation Property (Page 2237)
Name Property (Page 2228)
LimitMax Property (Page 2197)
Left Property (Page 2192)
Layer Property (Page 2160)
HiddenInput Property (Page 2141)
Height Property (Page 2139)
ForeFlashColorOn Property (Page 2133)
ForeFlashColorOff Property (Page 2132)
ForeColor Property (Page 2132)
FontUnderline Property (Page 2131)
FontSize Property (Page 2130)
FontName Property (Page 2129)
FontItalic Property (Page 2128)
FontBold Property (Page 2126)
FlashRateForeColor Property (Page 2123)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashForeColor Property (Page 2114)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)

4.4.7.3 L-Q

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:

```
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

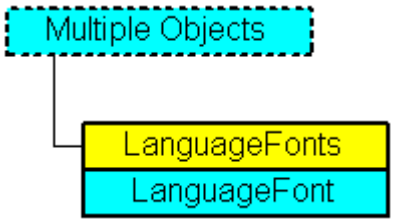
The LanguageFont object possesses the following properties:

See also

LanguageFonts Object (Listing) (Page 1890)
VBA Reference (Page 1664)
Underlined Property (Page 2324)
Size Property (Page 2291)
Parent Property (Page 2241)
LanguageID Property (Page 2158)
Italic Property (Page 2151)
FontFamily Property (Page 2127)
Bold Property (Page 2034)
Application Property (Page 2008)

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.

```
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 1889)

ItemByLcid Method (Page 1774)

Item Method (Page 1772)

VBA Reference (Page 1664)

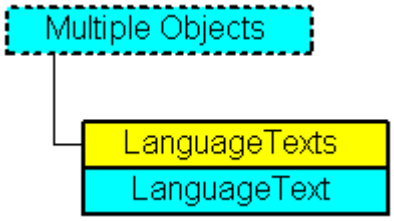
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

LanguageText Object

Description



Contains the multilingual labels for an object. The LanguageText object is an element of the LanguageTexts listing.

VBA Object Name

HMILanguageText

Usage

In the following example a German label and an English label will be assigned to the button called "myButton":

```
Sub AddLanguagesToButton()
'VBA280
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

LanguageTexts Object (Listing) (Page 1893)

Delete Method (Page 1745)

VBA Reference (Page 1664)

Parent Property (Page 2241)

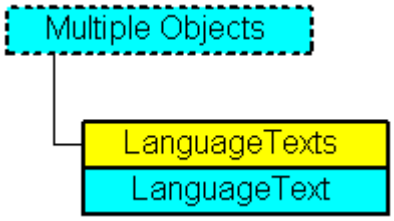
LanguageID Property (Page 2158)

DisplayText Property (Page 2094)

Application Property (Page 2008)

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
- LDTexts Property
- LDTooltipTexts 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 1892)

ItemByLcid Method (Page 1774)

Item Method (Page 1772)

VBA Reference (Page 1664)

Parent Property (Page 2241)

LDTooltipTexts Property (Page 2190)

LDTexts Property (Page 2190)

LDStatusTexts Property (Page 2188)

LDNames Property (Page 2187)

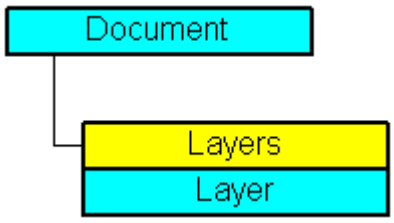
LDLabelTexts Property (Page 2186)

Count Property (Page 2082)

Application Property (Page 2008)

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:

```
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 2184)

VBA Reference (Page 1664)

Editing Layers with VBA (Page 1603)

Visible Property (Page 2405)

Number Property (Page 2230)

Name Property (Page 2228)

MinZoom Property (Page 2226)

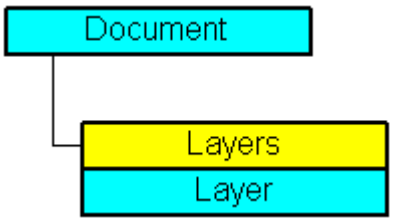
MaxZoom Property (Page 2208)

LDNames Property (Page 2187)

ActiveLayer Property (Page 1996)

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 1894)

Item Method (Page 1772)

VBA Reference (Page 1664)

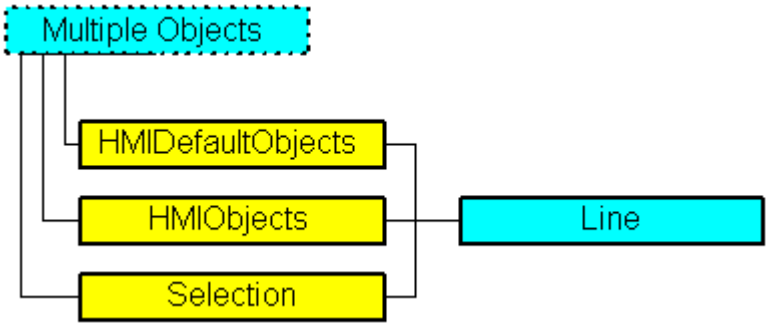
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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.
- HMIDefaultObjects: 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("Line1")
objLine.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

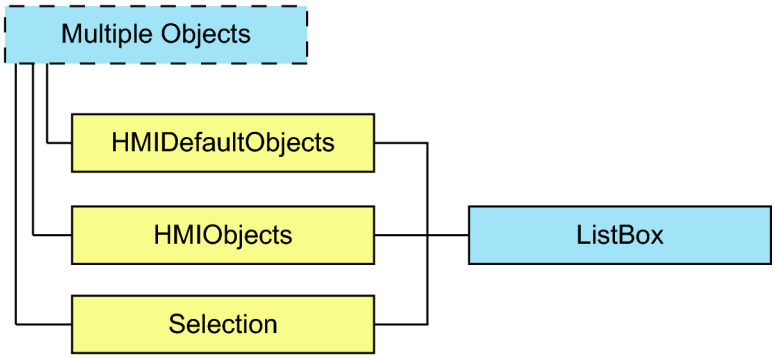
```
Sub ShowNameOfFirstSelectedObject()
'VBA287
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

AddHMIObject Method (Page 1719)
BorderBackColor Property (Page 2035)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
RotationAngle Property (Page 2274)
ReferenceRotationTop Property (Page 2271)
ReferenceRotationLeft Property (Page 2270)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Index Property (Page 2145)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashBorderColor Property (Page 2112)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderEndStyle Property (Page 2039)
BorderColor Property (Page 2036)
ActualPointTop Property (Page 1998)
ActualPointLeft Property (Page 1997)

ListBox object

Description



Represents the "ListBox" object. The ListBox 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

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.HMIObjects.AddHMIObject("ListBox", "HMIListBox")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```
Sub EditListBox()
'VBA830
Dim objListBox As HMIListBox
Set objListBox = ActiveDocument.HMIObjects("ListBox")
objListBox.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

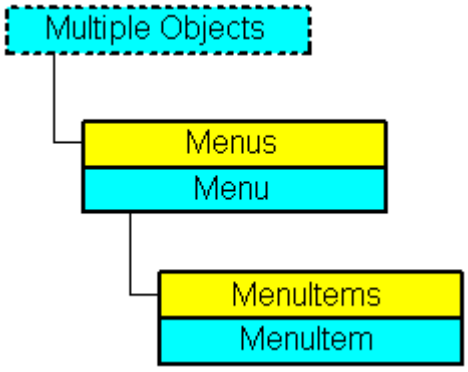
```
Sub ShowNameOfFirstSelectedObject()
'VBA831
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ObjectName Property (Page 2232)
Layer Property (Page 2160)
Left Property (Page 2192)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
NumberLines Property (Page 2231)
ForeColor Property (Page 2132)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackColor Property (Page 2017)
FillColor Property (Page 2104)
BorderStyle Property (Page 2043)
BorderWidth Property (Page 2045)
FillStyle Property (Page 2108)
GlobalShadow property (Page 2135)
FontName Property (Page 2129)
FontSize Property (Page 2130)
FontBold Property (Page 2126)
FontItalic Property (Page 2128)
FontUnderline Property (Page 2131)
AlignmentLeft Property (Page 2004)
Index Property (Page 2145)
Text Property (Page 2306)
Operation Property (Page 2237)
PasswordLevel Property (Page 2244)
Visible Property (Page 2405)
ToolTipText Property (Page 2310)
OperationMessage Property (Page 2238)
OperationReport Property (Page 2238)
SelIndex property (Page 2286)
SelText property (Page 2287)

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:

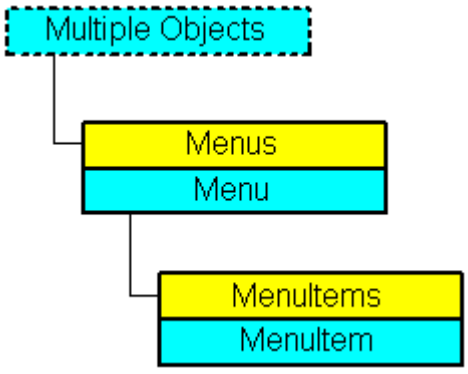
```
Sub DeleteMenu()
'VBA289
Dim objMenu As HMIMenu
Set objMenu = ActiveDocument.CustomMenus(1)
objMenu.Delete
End Sub
```

See also

Menus Object (Listing) (Page 1904)
Delete Method (Page 1745)
How to Create Picture-Specific Menus and Toolbars (Page 1602)
How to Create a New Application-Specific Menu (Page 1575)
VBA Reference (Page 1664)
Creating Customized Menus and Toolbars (Page 1573)
Visible Property (Page 2405)
StatusText Property (Page 2297)
Position Property (Page 2257)
Parent Property (Page 2241)
MenuItems Property (Page 2222)
LDStatusTexts Property (Page 2188)
LDLabelTexts Property (Page 2186)
Label Property (Page 2157)
Key Property (Page 2155)
Enabled Property (Page 2098)
Application Property (Page 2008)

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 1903)

Item Method (Page 1772)

InsertMenu Method (Page 1763)

How to Create Picture-Specific Menus and Toolbars (Page 1602)

How to Create a New Application-Specific Menu (Page 1575)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

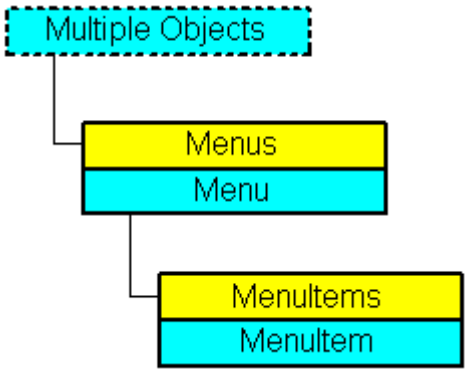
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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:

```
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:

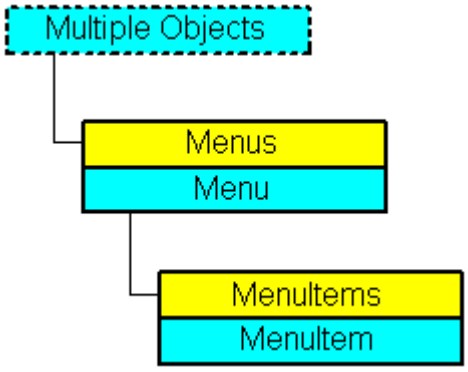
```
Sub DeleteMenuItem()
'VBA294
ActiveDocument.CustomMenus(1).MenuItems(1).Delete
End Sub
```

See also

Parent Property (Page 2241)
MenuItems Object (Listing) (Page 1909)
Delete Method (Page 1745)
Configuring Menus and Toolbars (Page 1572)
How to Assign VBA Macros to Menus and Toolbars (Page 1589)
How to Assign Help Texts to Menus and Toolbars (Page 1586)
How to Add a New Menu Item to the Menu (Page 1577)
VBA Reference (Page 1664)
Creating Customized Menus and Toolbars (Page 1573)
Visible Property (Page 2405)
Tag Property (Page 2303)
SubMenu Property (Page 2298)
StatusText Property (Page 2297)
ShortCut Property (Page 2289)
Position Property (Page 2257)
MenuItemType Property (Page 2223)
Macro Property (Page 2206)
LDStatusTexts Property (Page 2188)
LDLabelTexts Property (Page 2186)
Label Property (Page 2157)
Key Property (Page 2155)
Icon Property (Page 2144)
Enabled Property (Page 2098)
Checked Property (Page 2056)
Application Property (Page 2008)

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.

```
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:

```
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 1767)

MenuItem Object (Page 1906)

InsertSeparator Method (Page 1766)

InsertMenuItem Method (Page 1765)

How to Add a New Menu Item to the Menu (Page 1577)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

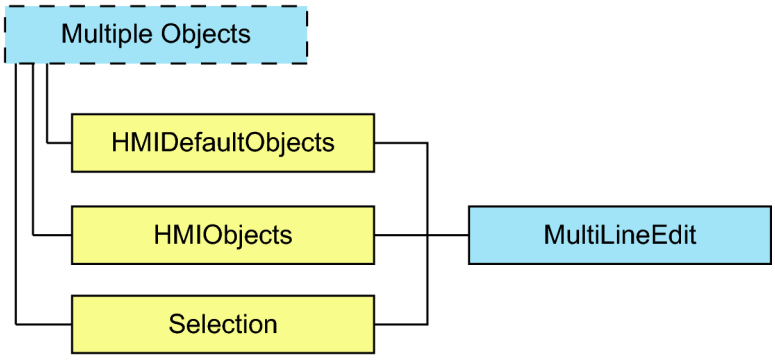
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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 2160)
Left Property (Page 2192)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BorderStyle Property (Page 2043)
BorderWidth Property (Page 2045)
BackColor Property (Page 2017)
FontName Property (Page 2129)
FontSize Property (Page 2130)
FontBold Property (Page 2126)
FontItalic Property (Page 2128)
FontUnderline Property (Page 2131)
ForeColor Property (Page 2132)
AlignmentLeft Property (Page 2004)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
Text Property (Page 2306)
Operation Property (Page 2237)
PasswordLevel Property (Page 2244)
Visible Property (Page 2405)
ToolTipText Property (Page 2310)
ObjectName Property (Page 2232)
GlobalShadow property (Page 2135)

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

Usage

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 & "' and '"
& strObjEnd & "'" & vbCrLf & "Connected points: " & vbCrLf & strObjStart & ": " & strStart
& vbCrLf & strObjEnd & ": " & 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 2312)

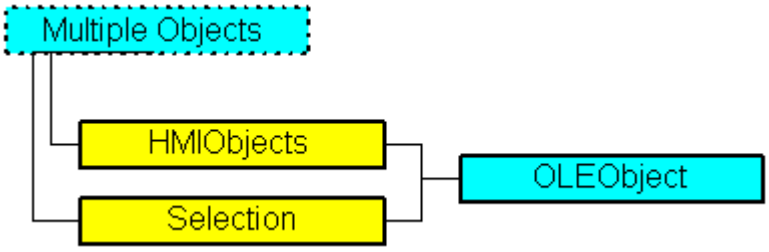
TopConnectedObjectName Property (Page 2312)

BottomConnectedConnectionPointIndex Property (Page 2047)

BottomConnectedObjectName Property (Page 2046)

OLEObject Object

Description



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:

```
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:

```
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:

```
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 1602)

Selection Object (Listing) (Page 1949)

HMIObjects Object (Listing) (Page 1884)

Delete Method (Page 1745)

AddOLEObject Method (Page 1722)

How to Create a New Application-Specific Toolbar (Page 1582)

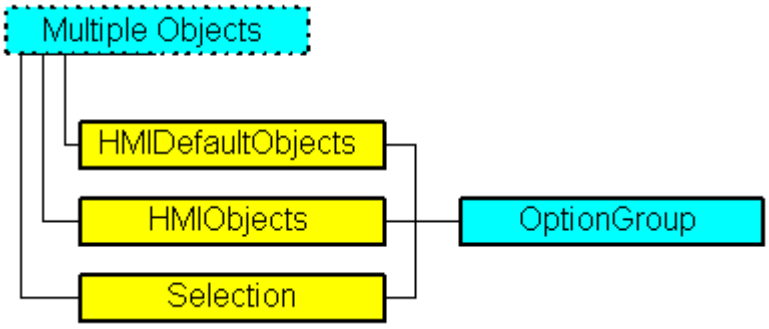
VBA Reference (Page 1664)

OLE Objects (Page 1615)

Application Property (Page 2008)

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:

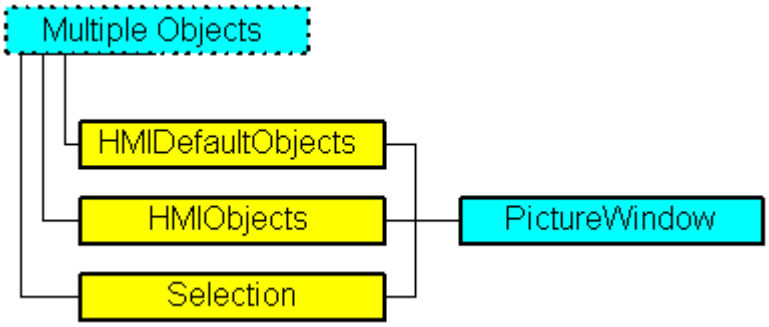
```
Sub ShowNameOfFirstSelectedObject()
'VBA303
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Left Property (Page 2192)
BorderStyle Property (Page 2043)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
Text Property (Page 2306)
Process Property (Page 2260)
PasswordLevel Property (Page 2244)
Orientation Property (Page 2239)
OperationMessage Property (Page 2238)
Operation Property (Page 2237)
Name Property (Page 2228)
Layer Property (Page 2160)
Index Property (Page 2145)
Height Property (Page 2139)
ForeFlashColorOn Property (Page 2133)
ForeFlashColorOff Property (Page 2132)
ForeColor Property (Page 2132)
FontUnderline Property (Page 2131)
FontSize Property (Page 2130)
FontName Property (Page 2129)
FontItalic Property (Page 2128)
FontBold Property (Page 2126)
FlashRateForeColor Property (Page 2123)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashForeColor Property (Page 2114)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)

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:

```
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:

```
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:

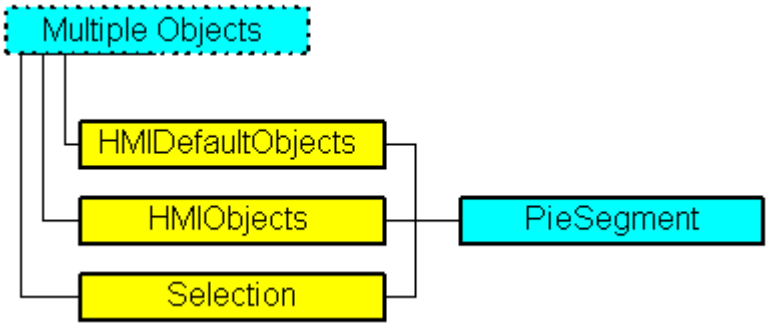
```
Sub ShowNameOfFirstSelectedObject()
'VBA306
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

MaximizeButton Property (Page 2208)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Zoom Property (Page 2414)
WindowBorder Property (Page 2409)
Width Property (Page 2408)
Visible Property (Page 2405)
UpdateCycle Property (Page 2327)
Top Property (Page 2311)
TagPrefix Property (Page 2304)
Sizeable Property (Page 2292)
ServerPrefix Property (Page 2288)
ScrollPositionY Property (Page 2282)
ScrollPositionX Property (Page 2282)
ScrollBars Property (Page 2281)
PictureName Property (Page 2252)
OnTop Property (Page 2236)
OffsetTop Property (Page 2235)
OffsetLeft Property (Page 2235)
Name Property (Page 2228)
Moveable Property (Page 2227)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
CloseButton Property (Page 2065)
CaptionText Property (Page 2054)
Caption Property (Page 2053)
AdaptSize Property (Page 2001)
AdaptPicture Property (Page 2000)

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.
- HMIDefaultObjects: 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:

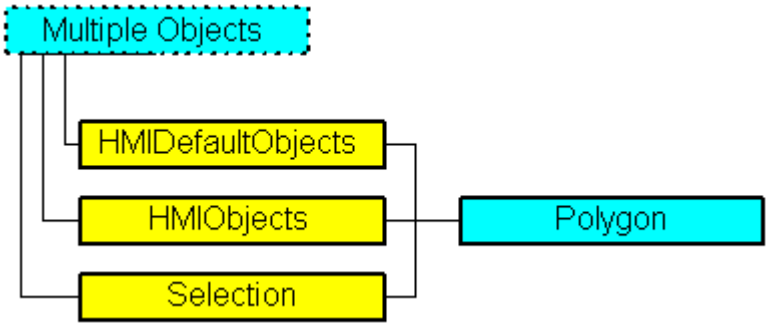
```
Sub ShowNameOfFirstSelectedObject()
'VBA309
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Filling Property (Page 2106)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
StartAngle Property (Page 2296)
Radius Property (Page 2268)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
FillColor Property (Page 2104)
EndAngle Property (Page 2098)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

Polygon Object

Description



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.
- HMIDefaultObjects: 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:

```
Sub AddPolygon()
'VBA310
Dim objPolygon As HMIPolygon
Set objPolygon = ActiveDocument.HMIObjects.AddHMIObject("Polygon", "HMIPolygon")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```
Sub EditPolygon()
'VBA311
Dim objPolygon As HMIPolygon
Set objPolygon = ActiveDocument.HMIObjects("Polygon")
objPolygon.BorderColor = RGB (255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing:

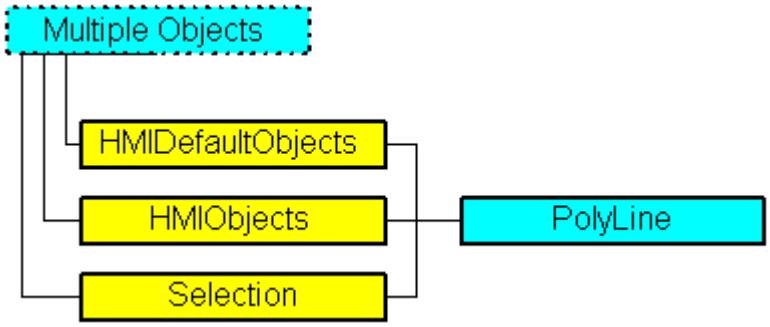
```
Sub ShowNameOfFirstSelectedObject()
'VBA312
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ToolTipText Property (Page 2310)
BorderBackColor Property (Page 2035)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
RotationAngle Property (Page 2274)
ReferenceRotationTop Property (Page 2271)
ReferenceRotationLeft Property (Page 2270)
PointCount Property (Page 2256)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Index Property (Page 2145)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)
ActualPointTop Property (Page 1998)
ActualPointLeft Property (Page 1997)

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:

```
Sub AddPolyLine()
'VBA313
Dim objPolyLine As HMIPolyLine
Set objPolyLine = ActiveDocument.HMIObjects.AddHMIObject("PolyLine1", "HMIPolyLine")
End Sub
```

Use "HMIObjects"(Index)" to return an object from the HMIObjects listing, where Index in this case identifies the object by name:

```
Sub EditPolyLine()
'VBA314
Dim objPolyLine As HMIPolyLine
Set objPolyLine = ActiveDocument.HMIObjects("PolyLine1")
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 "HMIDefaultObjects(Index)" to return an object from the HMIDefaultObjects Listing:

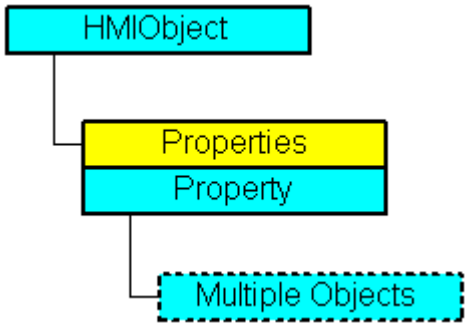
```
Sub EditDefaultPropertiesOfPolyLine()
'VBA316
Dim objPolyLine As HMIPolyLine
Set objPolyLine = Application.DefaultHMIObjects("HMIPolyLine")
objPolyLine.BorderColor = RGB(255, 255, 0)
End Sub
```

See also

HMIDefaultObjects Object (Listing) (Page 1878)
BorderEndStyle Property (Page 2039)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
RotationAngle Property (Page 2274)
ReferenceRotationTop Property (Page 2271)
ReferenceRotationLeft Property (Page 2270)
PointCount Property (Page 2256)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Index Property (Page 2145)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashBorderColor Property (Page 2112)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
ActualPointTop Property (Page 1998)
ActualPointLeft Property (Page 1997)

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

Item Method (Page 1772)

VBA Reference (Page 1664)

Editing Objects with VBA (Page 1606)

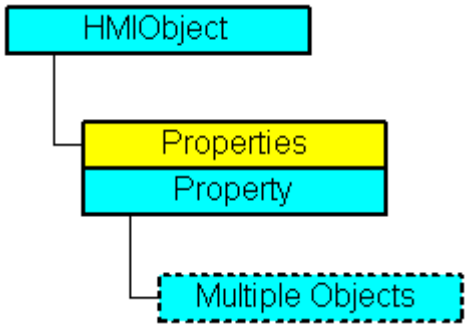
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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 2092)

Properties Object (Listing) (Page 1931)

DeleteDynamic Method (Page 1747)

CreateDynamic Method (Page 1742)

VBA Reference (Page 1664)

Creating Dynamics with VBA (Page 1631)

Editing Objects with VBA (Page 1606)

Value Property (Page 2331)

Type Property (Page 2315)

Parent Property (Page 2241)

Name Property (Page 2228)

IsDynamicable Property (Page 2150)

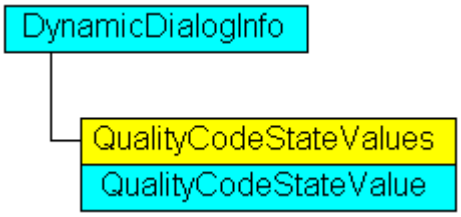
Events Property (Page 2099)

Dynamic Property (Page 2095)

Application Property (Page 2008)

QualityCodeStateValue Object

Description



Represents the quality code of a tag which is assigned in the dynamic dialog and used for dynamization.

VBA Object Name

HMIQualityCodeStateValue

Object properties

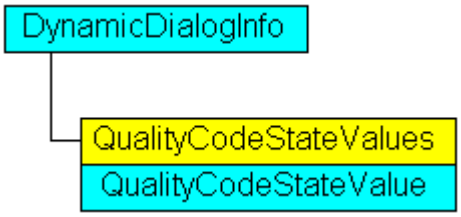
The object QualityCodeStateValue has the following properties:

See also

VALUE_BAD_CONFERROR Property (Page 2338)
VBA Reference (Page 1664)
VarName Property (Page 2403)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
Parent Property (Page 2241)
Application Property (Page 2008)

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

Usage

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 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:

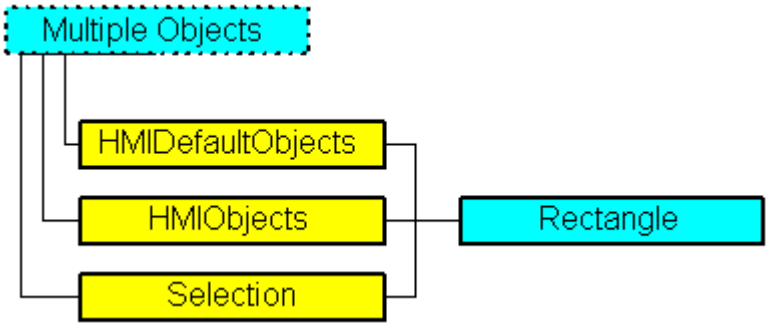
See also

VALUE_UNCERT_MAINTDEM Property (Page 2387)
VBA Reference (Page 1664)
VarName Property (Page 2403)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
Parent Property (Page 2241)
Item Property (Page 2152)
Count Property (Page 2082)
Application Property (Page 2008)

4.4.7.4 R-Z

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:

```
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:

```
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:

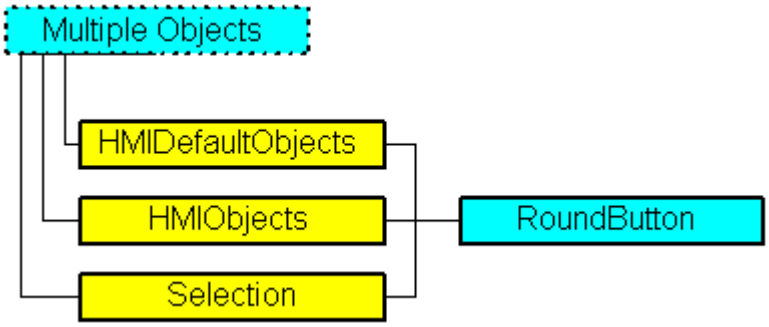
```
Sub ShowNameOfFirstSelectedObject()
'VBA322
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Name Property (Page 2228)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

RoundButton Object

Description



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:

```
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:

```
Sub EditRoundButton()
'VBA324
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects("Roundbutton1")
objRoundButton.BorderColor = RGB(255, 0, 0)
End Sub
```

Use "Selection"(Index) to return an object from the Selection listing.:

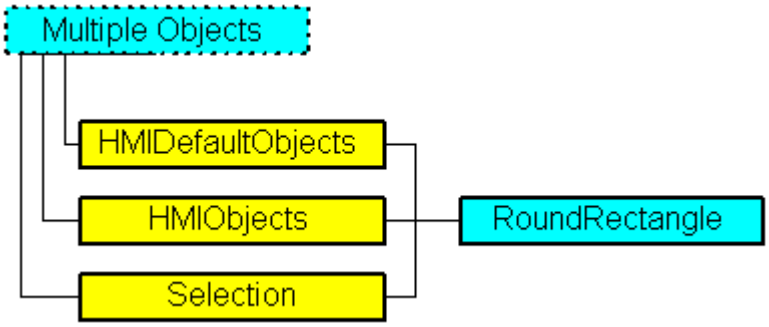
```
Sub ShowNameOfFirstSelectedObject()
'VBA325
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ToolTipText Property (Page 2310)
FlashBackColor Property (Page 2111)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
Toggle Property (Page 2307)
Radius Property (Page 2268)
Pressed Property (Page 2260)
PicUpUseTransColor Property (Page 2255)
PicUpTransparent Property (Page 2254)
PicUpReferenced Property (Page 2254)
PictureUp Property (Page 2253)
PictureDown Property (Page 2251)
PictureDeactivated Property (Page 2251)
PicDownUseTransColor Property (Page 2249)
PicDownTransparent Property (Page 2248)
PicDownReferenced Property (Page 2247)
PicDeactUseTransColor Property (Page 2247)
PicDeactTransparent Property (Page 2246)
PicDeactReferenced-Eigenschaft (Page 2245)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)

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:

```
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:

```
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:

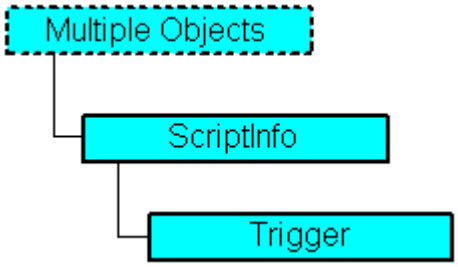
```
Sub ShowNameOfFirstSelectedObject()
'VBA328
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Width Property (Page 2408)
BorderBackColor Property (Page 2035)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
RoundCornerWidth Property (Page 2275)
RoundCornerHeight Property (Page 2275)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BackFlashColorOn Property (Page 2022)
BackFlashColorOff Property (Page 2021)
BackColor Property (Page 2017)

ScriptInfo Object

Description



Represents 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 2264)

Delete Method (Page 1745)

VBA Reference (Page 1664)

Creating Dynamics with VBA (Page 1631)

Trigger Property (Page 2314)

SourceCode Property (Page 2295)

ScriptType Property (Page 2280)

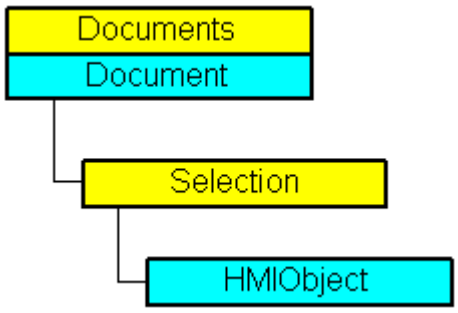
Parent Property (Page 2241)

Compiled Property (Page 2080)

Application Property (Page 2008)

Selection 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:

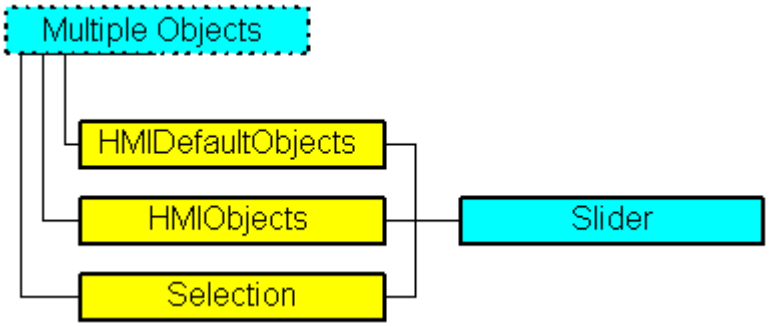
```
Sub SelectAllObjects()
'VBA332
ActiveDocument.Selection.SelectAll
End Sub
```

See also

HMIObjects Object (Listing) (Page 1884)
AlignTop Method (Page 1727)
HMIDefaultObjects Object (Listing) (Page 1878)
BringToFront Method (Page 1730)
SendToBack Method (Page 1795)
SelectAll Method (Page 1794)
SameWidthAndHeight Method (Page 1790)
SameWidth Method (Page 1788)
SameHeight Method (Page 1787)
Rotate Method (Page 1786)
Remove Method (Page 1784)
ForwardOneLevel Method (Page 1759)
BackwardOneLevel Method (Page 1729)
MoveSelection Method (Page 1779)
Item Method (Page 1772)
FlipVertically Method (Page 1758)
FlipHorizontally Method (Page 1757)
EvenlySpaceVertically Method (Page 1752)
EvenlySpaceHorizontally Method (Page 1751)
DuplicateSelection Method (Page 1750)
DeselectAll Method (Page 1748)
DeleteAll Method (Page 1746)
CreateGroup Method (Page 1743)
CreateCustomizedObject Method (Page 1741)
CopySelection Method (Page 1738)
CenterVertically Method (Page 1733)
CenterHorizontally Method (Page 1732)
AlignRight Method (Page 1726)
AlignLeft Method (Page 1725)
AlignBottom Method (Page 1724)
How to Edit a Customized Object with VBA (Page 1628)
How to Edit Group Objects with VBA (Page 1620)
How to edit Default objects, Smart objects, Windows objects and Tube objects (Page 1610)
VBA Reference (Page 1664)
Customized Objects (Page 1626)
Group Objects (Page 1618)
Default objects, Smart objects, Windows objects and Tube objects (Page 1609)
Editing Objects with VBA (Page 1606)
Parent Property (Page 2241)
Count Property (Page 2082)

Slider object

Description



Represents 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:

```
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:

```
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:

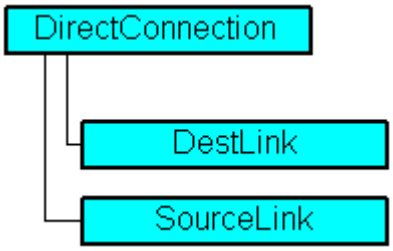
```
Sub ShowNameOfFirstSelectedObject()
'VBA335
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

OperationReport Property (Page 2238)
BorderFlashColorOff Property (Page 2040)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
SmallChange Property (Page 2292)
Process Property (Page 2260)
PasswordLevel Property (Page 2244)
OperationMessage Property (Page 2238)
Operation Property (Page 2237)
Name Property (Page 2228)
Min Property (Page 2225)
Max Property (Page 2207)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
ExtendedOperation Property (Page 2103)
Direction Property (Page 2091)
ColorTop Property (Page 2075)
ColorBottom Property (Page 2068)
ButtonColor Property (Page 2052)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderColor Property (Page 2036)

SourceLink Object

Description



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 1845)

VBA Reference (Page 1664)

Type Property (Page 2315)

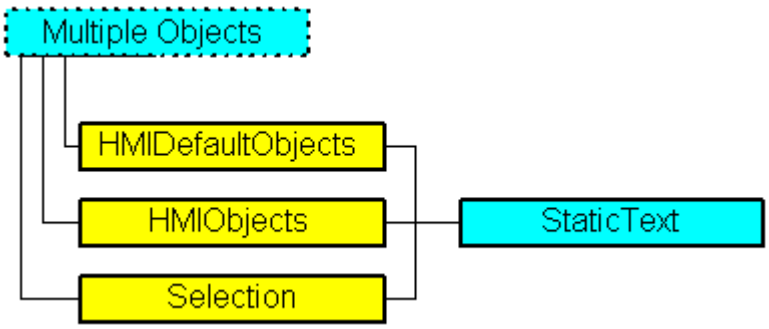
SourceLink Property (Page 2293)

ObjectName Property (Page 2232)

AutomationName Property (Page 2011)

StaticText Object

Description



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:

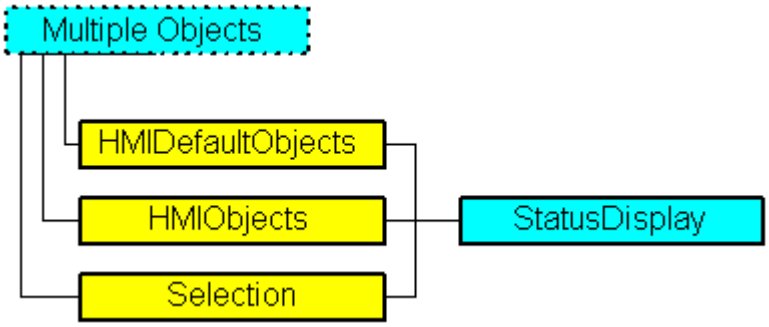
```
Sub ShowNameOfFirstSelectedObject()
'VBA339
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Selection Object (Listing) (Page 1949)
FontBold Property (Page 2126)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
Text Property (Page 2306)
PasswordLevel Property (Page 2244)
Orientation Property (Page 2239)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Height Property (Page 2139)
ForeFlashColorOn Property (Page 2133)
ForeFlashColorOff Property (Page 2132)
ForeColor Property (Page 2132)
FontUnderline Property (Page 2131)
FontSize Property (Page 2130)
FontName Property (Page 2129)
FontItalic Property (Page 2128)
FlashRateForeColor Property (Page 2123)
FlashRateBorderColor Property (Page 2120)
FlashRateBackColor Property (Page 2119)
FlashForeColor Property (Page 2114)
FlashBorderColor Property (Page 2112)
FlashBackColor Property (Page 2111)
FillStyle Property (Page 2108)
FillingIndex Property (Page 2107)
Filling Property (Page 2106)
FillColor Property (Page 2104)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)

StatusDisplay Object

Description



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:

```
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:

```
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:

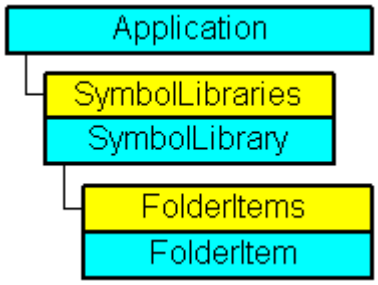
```
Sub ShowNameOfFirstSelectedObject()
'VBA342
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ToolTipText Property (Page 2310)
BasePicReferenced Property (Page 2027)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Width Property (Page 2408)
Visible Property (Page 2405)
Top Property (Page 2311)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
Name Property (Page 2228)
Left Property (Page 2192)
Layer Property (Page 2160)
Index Property (Page 2145)
Height Property (Page 2139)
FlashRateFlashPic Property (Page 2122)
FlashRateBorderColor Property (Page 2120)
FlashPicUseTransColor Property (Page 2117)
FlashPicture Property (Page 2116)
FlashPicTransColor Property (Page 2116)
FlashPicReferenced Property (Page 2115)
FlashFlashPicture Property (Page 2114)
FlashBorderColor Property (Page 2112)
BorderWidth Property (Page 2045)
BorderStyle Property (Page 2043)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
BorderColor Property (Page 2036)
BorderBackColor Property (Page 2035)
BasePicUseTransColor Property (Page 2029)
BasePicture Property (Page 2028)
BasePicTransColor Property (Page 2027)

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:

```
Sub ShowFirstObjectOfCollection()
'VBA343
Dim strName As String
strName = Application.SymbolLibraries(1).Name
MsgBox strName
End Sub
```

See also

SymbolLibraries Object (Listing) (Page 1963)

GetItemByPath Method (Page 1760)

FindByDisplayName Method (Page 1756)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

Parent Property (Page 2241)

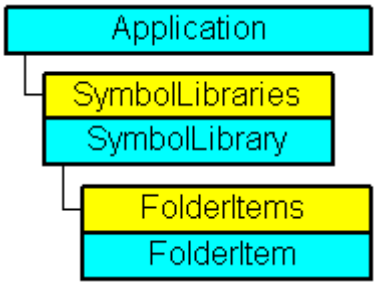
Name Property (Page 2228)

FolderItems Property (Page 2125)

Application Property (Page 2008)

SymbolLibraries Object (Listing)

Description



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 1962)

Item Method (Page 1772)

VBA Reference (Page 1664)

Accessing the component library with VBA (Page 1593)

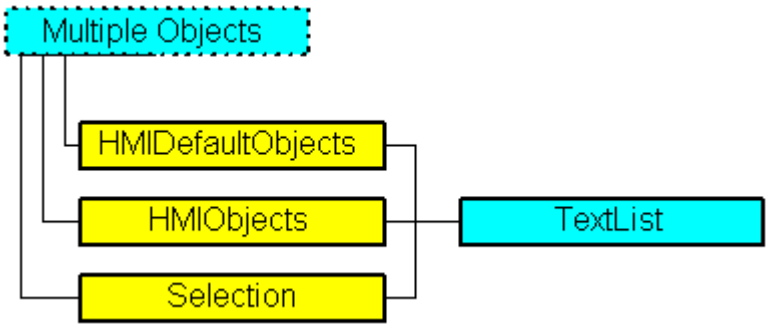
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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:

```
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:

```
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:

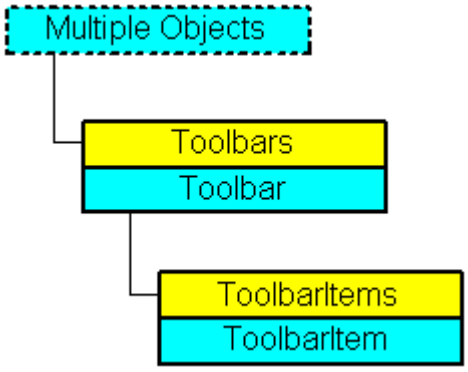
```
Sub ShowNameOfFirstSelectedObject()
'VBA347
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name of the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

Width Property (Page 2408)
ForeFlashColorOn Property (Page 2133)
BitNumber Property (Page 2032)
Selection Object (Listing) (Page 1949)
HMIObjects Object (Listing) (Page 1884)
HMIDefaultObjects Object (Listing) (Page 1878)
AddHMIObject Method (Page 1719)
VBA Reference (Page 1664)
Editing Objects with VBA (Page 1606)
Visible Property (Page 2405)
UnselTextColor Property (Page 2326)
UnselBGColor Property (Page 2325)
Top Property (Page 2311)
ToolTipText Property (Page 2310)
SelTextColor Property (Page 2287)
SelBGColor Property (Page 2284)
PasswordLevel Property (Page 2244)
OutputValue Property (Page 2241)
Orientation Property (Page 2239)
OperationReport Property (Page 2238)
OperationMessage Property (Page 2238)
Operation Property (Page 2237)
NumberLines Property (Page 2231)
Name Property (Page 2228)
ListType Property (Page 2198)
Left Property (Page 2192)
Layer Property (Page 2160)
LanguageSwitch Property (Page 2159)
ItemBorderWidth Property (Page 2155)
ItemBorderStyle Property (Page 2154)
ItemBorderColor Property (Page 2153)
ItemBorderBackColor Property (Page 2152)
Height Property (Page 2139)
ForeFlashColorOff Property (Page 2132)
ForeColor Property (Page 2132)
FontUnderline Property (Page 2131)
FontSize Property (Page 2130)
FontName Property (Page 2129)
FontItalic Property (Page 2128)
FontBold Property (Page 2126)

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:

```
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:

```
Sub DeleteToolbar()
'VBA349
Dim objToolbar As HMIToolbar
Set objToolbar = ActiveDocument.CustomToolbars(1)
objToolbar.Delete
End Sub
```

See also

Key Property (Page 2155)

Toolbars Object (Listing) (Page 1968)

Delete Method (Page 1745)

How to Create Picture-Specific Menus and Toolbars (Page 1602)

How to Create a New Application-Specific Toolbar (Page 1582)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

Visible Property (Page 2405)

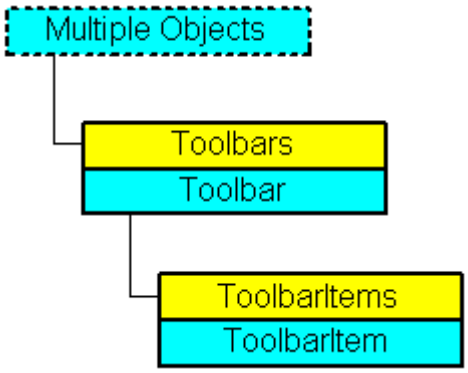
ToolbarItems Property (Page 2309)

Parent Property (Page 2241)

Application Property (Page 2008)

Toolbars Object (Listing)

Description



A listing of the Toolbar objects that represent all the user-defined toolbars in the Graphics Designer.

VBA Object Name

HMICustomToolbars

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 1967)

Item Method (Page 1772)

Add Method (Page 1705)

How to Create Picture-Specific Menus and Toolbars (Page 1602)

How to Create a New Application-Specific Toolbar (Page 1582)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

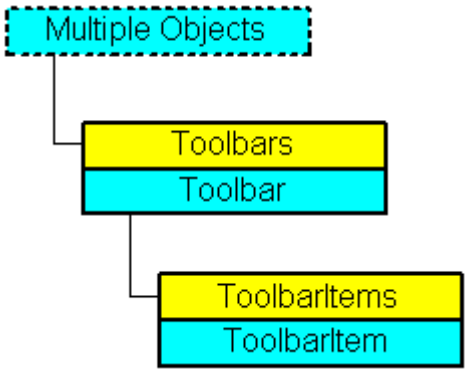
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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:

```
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:

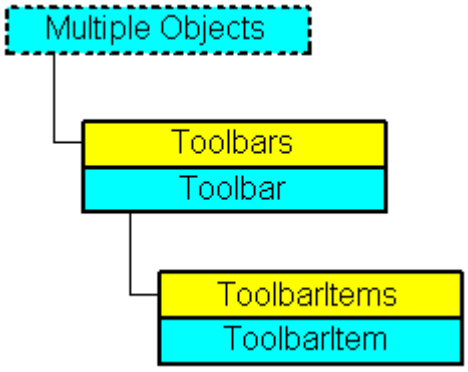
```
Sub DeleteToolbarItem()
'VBA354
ActiveDocument.CustomToolbars(1).ToolbarItems(1).Delete
End Sub
```

See also

Macro Property (Page 2206)
ToolbarItems Object (Listing) (Page 1973)
Delete Method (Page 1745)
Configuring Menus and Toolbars (Page 1572)
How to Assign VBA Macros to Menus and Toolbars (Page 1589)
How to Assign Help Texts to Menus and Toolbars (Page 1586)
How to Add a New Icon to the Toolbar (Page 1584)
VBA Reference (Page 1664)
Creating Customized Menus and Toolbars (Page 1573)
Visible Property (Page 2405)
Type Property (Page 2315)
ToolTipText Property (Page 2310)
Tag Property (Page 2303)
StatusText Property (Page 2297)
ShortCut Property (Page 2289)
Position Property (Page 2257)
Parent Property (Page 2241)
LDTooltipTexts Property (Page 2190)
LDStatusTexts Property (Page 2188)
Key Property (Page 2155)
Icon Property (Page 2144)
Enabled Property (Page 2098)
Application Property (Page 2008)

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 1970)

InsertToolbarItem Method (Page 1769)

InsertSeparator Method (Page 1766)

InsertFromMenuItem Method (Page 1762)

How to Add a New Icon to the Toolbar (Page 1584)

VBA Reference (Page 1664)

Creating Customized Menus and Toolbars (Page 1573)

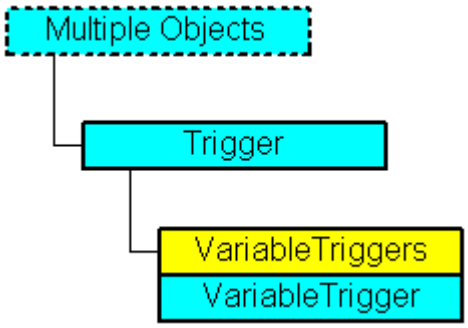
Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

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 1745)

VBA Reference (Page 1664)

VariableTriggers Property (Page 2402)

Type Property (Page 2315)

Trigger Property (Page 2314)

Parent Property (Page 2241)

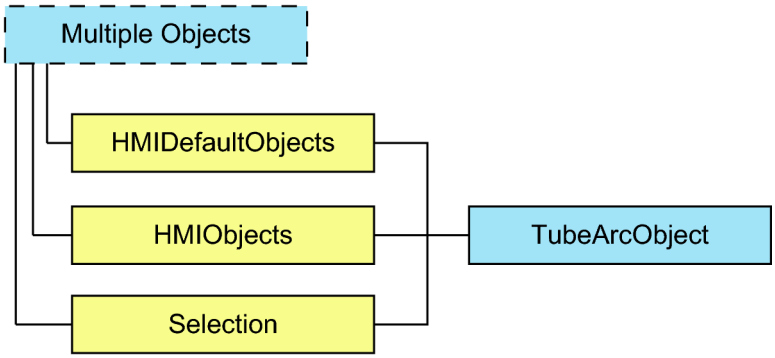
Name Property (Page 2228)

CycleType Property (Page 2088)

Application Property (Page 2008)

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:

```
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:

```
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:

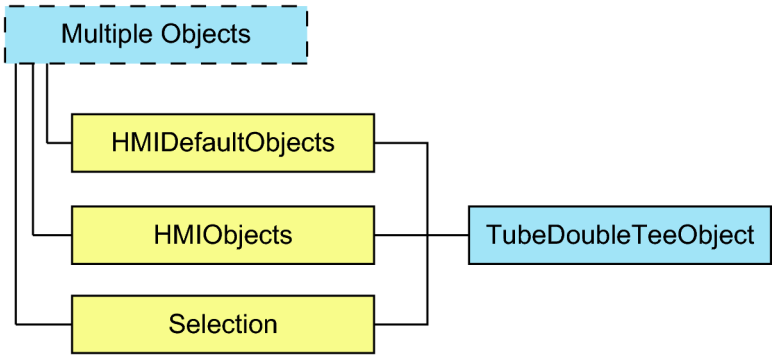
```
Sub ShowNameOfFirstSelectedObject()
'VBA837
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ObjectName Property (Page 2232)
Left Property (Page 2192)
Layer Property (Page 2160)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
BorderColor Property (Page 2036)
BorderWidth Property (Page 2045)
ToolTipText Property (Page 2310)
Visible Property (Page 2405)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
FlashRateBorderColor Property (Page 2120)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
FlashBorderColor Property (Page 2112)
Transparency property (Page 2313)
GlobalShadow property (Page 2135)
GlobalColorScheme property (Page 2134)
StartAngle Property (Page 2296)
EndAngle Property (Page 2098)
RadiusHeight Property (Page 2268)
RadiusWidth Property (Page 2269)

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:

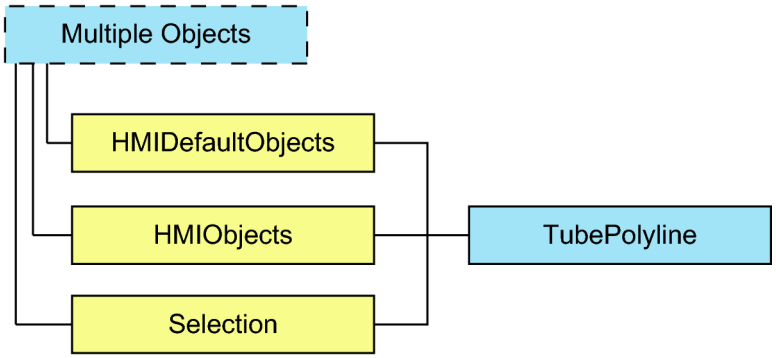
```
Sub ShowNameOfFirstSelectedObject()
'VBA840
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ObjectName Property (Page 2232)
Left Property (Page 2192)
Layer Property (Page 2160)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
BorderColor Property (Page 2036)
BorderWidth Property (Page 2045)
ToolTipText Property (Page 2310)
Visible Property (Page 2405)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
FlashRateBorderColor Property (Page 2120)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
FlashBorderColor Property (Page 2112)
Transparency property (Page 2313)
GlobalShadow property (Page 2135)
GlobalColorScheme property (Page 2134)

TubePolyline object

Description



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:

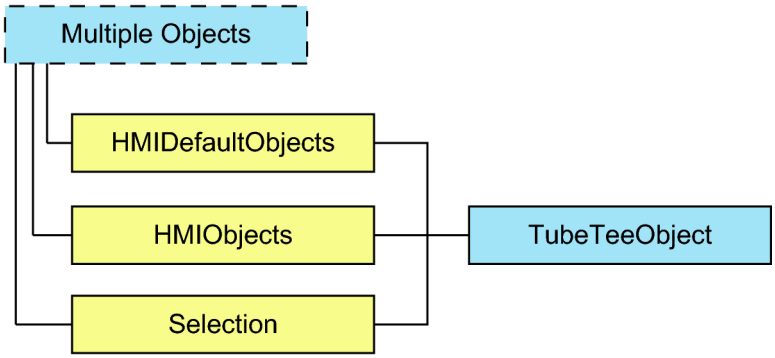
```
Sub ShowNameOfFirstSelectedObject()
'VBA843
'Select all objects in the picture:
ActiveDocument.Selection.SelectAll
'Get the name from the first object of the selection:
MsgBox ActiveDocument.Selection(1).ObjectName
End Sub
```

See also

ObjectName Property (Page 2232)
Left Property (Page 2192)
Layer Property (Page 2160)
Top Property (Page 2311)
Width Property (Page 2408)
Height Property (Page 2139)
BorderColor Property (Page 2036)
BorderWidth Property (Page 2045)
ToolTipText Property (Page 2310)
Visible Property (Page 2405)
PasswordLevel Property (Page 2244)
Operation Property (Page 2237)
FlashRateBorderColor Property (Page 2120)
BorderFlashColorOn Property (Page 2042)
BorderFlashColorOff Property (Page 2040)
FlashBorderColor Property (Page 2112)
Transparency property (Page 2313)
GlobalShadow property (Page 2135)
GlobalColorScheme property (Page 2134)
PointCount Property (Page 2256)
ActualPointLeft Property (Page 1997)
ActualPointTop Property (Page 1998)
Index Property (Page 2145)
LineJoinStyle property (Page 2198)

TubeTeeObject object

Description



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:

```
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:

```
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:

```
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 2232)

Left Property (Page 2192)

Layer Property (Page 2160)

Top Property (Page 2311)

Width Property (Page 2408)

Height Property (Page 2139)

BorderColor Property (Page 2036)

BorderWidth Property (Page 2045)

ToolTipText Property (Page 2310)

Visible Property (Page 2405)

PasswordLevel Property (Page 2244)

Operation Property (Page 2237)

FlashRateBorderColor Property (Page 2120)

BorderFlashColorOn Property (Page 2042)

BorderFlashColorOff Property (Page 2040)

FlashBorderColor Property (Page 2112)

Transparency property (Page 2313)

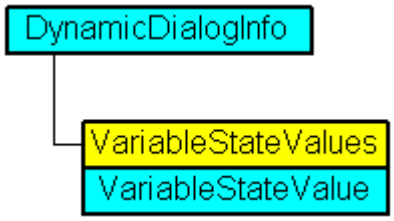
GlobalShadow property (Page 2135)

GlobalColorScheme property (Page 2134)

RotationAngle Property (Page 2274)

VariableStateValue Object

Description



Represents the state of a tag, the value of which is assigned in the Dynamic dialog and used

VBA Object Name

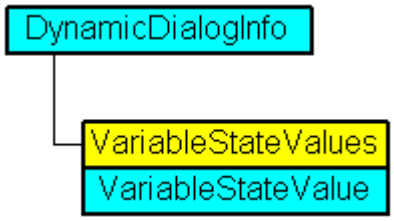
HMIVariableStateValue

See also

VALUE_SERVERDOWN Property (Page 2373)
VBA Reference (Page 1664)
VarName Property (Page 2403)
VALUE_TIMEOUT Property (Page 2376)
VALUE_STARTUP_VALUE Property (Page 2375)
VALUE_NOT_ESTABLISHED Property (Page 2372)
VALUE_MIN_RANGE Property (Page 2370)
VALUE_MIN_LIMIT Property (Page 2369)
VALUE_MAX_RANGE Property (Page 2367)
VALUE_MAX_LIMIT Property (Page 2365)
VALUE_INVALID_KEY Property (Page 2361)
VALUE_HARDWARE_ERROR Property (Page 2358)
VALUE_HANDSHAKE_ERROR Property (Page 2356)
VALUE_CONVERSION_ERROR Property (Page 2354)
VALUE_ADDRESS_ERROR Property (Page 2333)
VALUE_ACCESS_FAULT Property (Page 2331)
Parent Property (Page 2241)
Application Property (Page 2008)

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:

```
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
```

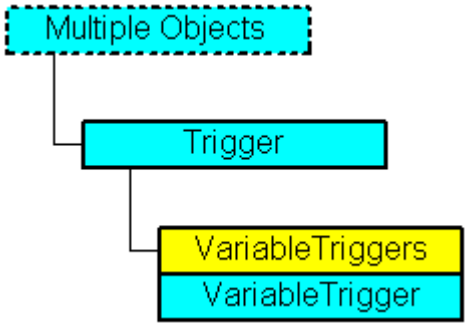
```
.VALUE_MIN_RANGE = 120
.VALUE_NOT_ESTABLISHED = 130
.VALUE_SERVERDOWN = 140
.VALUE_STARTUP_VALUE = 150
.VALUE_TIMEOUT = 160
End With
End Sub
```

See also

VALUE_MAX_RANGE Property (Page 2367)
VBA Reference (Page 1664)
VarName Property (Page 2403)
VALUE_TIMEOUT Property (Page 2376)
VALUE_STARTUP_VALUE Property (Page 2375)
VALUE_SERVERDOWN Property (Page 2373)
VALUE_NOT_ESTABLISHED Property (Page 2372)
VALUE_MIN_RANGE Property (Page 2370)
VALUE_MIN_LIMIT Property (Page 2369)
VALUE_MAX_LIMIT Property (Page 2365)
VALUE_INVALID_KEY Property (Page 2361)
VALUE_HARDWARE_ERROR Property (Page 2358)
VALUE_HANDSHAKE_ERROR Property (Page 2356)
VALUE_CONVERSION_ERROR Property (Page 2354)
VALUE_ADDRESS_ERROR Property (Page 2333)
VALUE_ACCESS_FAULT Property (Page 2331)
Parent Property (Page 2241)
Item Property (Page 2152)
Application Property (Page 2008)

VariableTrigger Object

Description



Represents a tag trigger.

VBA Object Name

HMIVariableTrigger

Usage

Use the VariableTrigger object in order to edit or delete an existing tag trigger. "In this example a circle property "Top" will be made dynamic with the aid of the tag Otto:

```
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 1745)

VariableTriggers Object (Listing) (Page 1990)

VBA Reference (Page 1664)

VariableTriggers Property (Page 2402)

Type Property (Page 2315)

Parent Property (Page 2241)

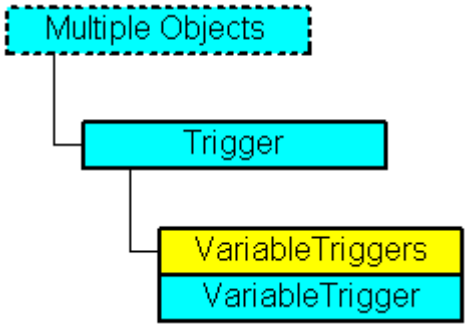
Name Property (Page 2228)

CycleType Property (Page 2088)

Application Property (Page 2008)

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:

```
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 1712)

VBA Reference (Page 1664)

Parent Property (Page 2241)

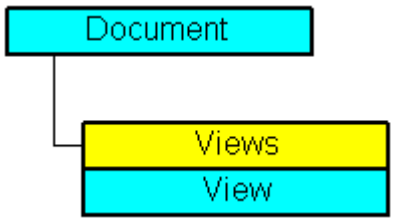
Item Property (Page 2152)

Count Property (Page 2082)

Application Property (Page 2008)

View Object

Description



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:

```
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:

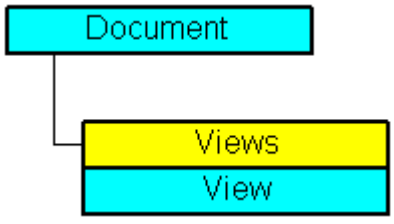
```
Sub AddView()
'VBA362
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Activate
End Sub
```

See also

Height Property (Page 2139)
Views Object (Listing) (Page 1993)
SetCSLayerVisible Method (Page 1797)
PrintDocument Method (Page 1782)
IsCSLayerVisible Method (Page 1770)
Delete Method (Page 1745)
Add Method (Views Listing) (Page 1713)
Activate Method (Page 1704)
VBA Reference (Page 1664)
Editing a Copy of a Picture with VBA (Page 1605)
Editing Layers with VBA (Page 1603)
ExtendedZoomingEnable Property (Page 2103)
Zoom Property (Page 2414)
WindowState Property (Page 2412)
Width Property (Page 2408)
Top Property (Page 2311)
ScrollPosY Property (Page 2284)
ScrollPosX Property (Page 2283)
Parent Property (Page 2241)
Left Property (Page 2192)
IsActive Property (Page 2148)
Application Property (Page 2008)
ActiveLayer Property (Page 1996)

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:

```
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:

```
Sub AddViewToActiveDocument()
'VBA364
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Activate
End Sub
```

See also

Item Method (Page 1772)

View Object (Page 1991)

Add Method (Page 1705)

VBA Reference (Page 1664)

Parent Property (Page 2241)

Count Property (Page 2082)

Application Property (Page 2008)

4.4.8 Properties

4.4.8.1 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:

```
Sub CreateVBActionToClickedEvent()
'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 1810)

AddAction Method (Page 1713)

Configuring Event-Driven Actions with VBA (Page 1644)

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 1850)

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 1991)

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:

```
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 2256)

Index Property (Page 2145)

ActualPointTop Property (Page 1998)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

Line Object (Page 1897)

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:

```
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 2256)

Index Property (Page 2145)

ActualPointLeft Property (Page 1997)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

Line Object (Page 1897)

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.

```
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 1916)

TextList Object (Page 1964)

StaticText Object (Page 1956)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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:

```
Sub PictureWindowConfig()
'VBA373
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 1919)

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:

```
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 1919)

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
'Set barcolor to "yellow"
.ColorAlarmHigh = RGB(255, 255, 0)
'set upper limit to "50"
.AlarmHigh = 50
End With
End Sub
```

See also

TypeAlarmHigh Property (Page 2316)

ColorAlarmHigh Property (Page 2066)

CheckAlarmHigh Property (Page 2055)

BarGraph Object (Page 1819)

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 2317)

ColorAlarmLow Property (Page 2067)

CheckAlarmLow Property (Page 2056)

BarGraph Object (Page 1819)

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.

Display	Assigned Value
Right or bottom	TRUE
Left or top	FALSE

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 2278)

Direction Property (Page 2091)

BarGraph Object (Page 1819)

AlignmentLeft Property

Description

Defines or returns the horizontal alignment of the text. Value range from 0 to 2.

Horizontal Alignment	Assigned Value
Left	0
Centered	1
Right	2

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:

```
Sub IOFieldConfiguration()
'VBA378
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.AlignmentLeft = 1
End With
End Sub
```

Related topics

See also

AlignmentTop Property (Page 2005)
TextList Object (Page 1964)
StaticText Object (Page 1956)
OptionGroup Object (Page 1916)
GroupDisplay Object (Page 1874)
IOField Object (Page 1886)
CheckBox Object (Page 1827)
Button Object (Page 1824)

AlignmentTop Property

Description

Defines or returns the vertical alignment of the text. Value range from 0 to 2.

Horizontal Alignment	Assigned Value
Up	0
Centered	1
Down	2

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 2004)

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

GroupDisplay Object (Page 1874)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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

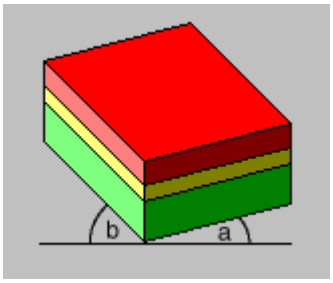
DynamicDialog Object (Page 1851)

AnalogResultInfos Object (Listing) (Page 1813)

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.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

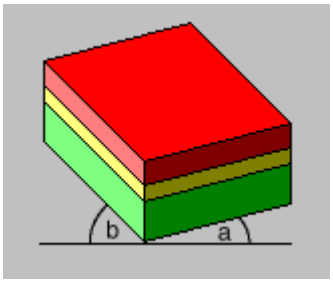
AngleBeta Property (Page 2007)

3DBarGraph Object (Page 1805)

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 2006)

3DBarGraph Object (Page 1805)

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:

```
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 1814)

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:

```
Sub ShowApplicationDataPath()
'VBA383
MsgBox Application.ApplicationDataPath
End Sub
```

See also

Application Property (Page 2008)

Application Object (Page 1814)

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 2198)

TextList Object (Page 1964)

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.

```
Sub IOFieldConfiguration()
'VBA385
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.AssumeOnExit = True
End With
End Sub
```

See also

TextList Object (Page 1964)

IOField Object (Page 1886)

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.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.AssumeOnFull = True
End With
```

```
End Sub
```

See also

OutputFormat Property (Page 2240)

DataFormat Property (Page 2089)

IOField Object (Page 1886)

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)

Type Property	AutomationName Property	ObjectName Property
hmiSourceTypeConstant	--	Name of the constant (e.g. the picture name)
hmiSourceTypeProperty	Property of the source object (e.g. "Top")	Name of the source object (e.g. "Rectangle_A")
hmiSourceTypePropertyOfThisObject	--	--
hmiSourceTypeVariableDirect	--	Tag name
hmiSourceTypeVariableIndirect	--	Tag name

Destination object type (DestinationLink Property)

Type Property	AutomationName Property	ObjectName Property
hmiDestTypeProperty	Property of the destination object (e.g. "Left")	Name of the destination object (e.g. "Rectangle_A")
hmiDestTypePropertyOfThisObject	--	--
hmiDestTypePropertyOfActualWindow	Property of the destination object (e.g. "Left")	--
hmiDestTypeVariableDirect	--	Tag name
hmiDestTypeVariableIndirect	--	Tag name
hmiDestTypeDirectMessage	--	Tag name
hmiDestTypeIndirectMessage	--	Tag name

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
```

See also

DestinationLink Property (Page 2090)

Type Property (Page 2315)

SourceLink Property (Page 2293)

ObjectName Property (Page 2232)

SourceLink Object (Page 1955)

DestLink Object (Page 1843)

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):

```
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
```

See also

LanguageName Property (Page 2158)

LanguageID Property (Page 2158)

How to Assign Help Texts to Menus and Toolbars (Page 1586)

How to Create Menus in Multiple Languages (Page 1580)

Language-Dependent Configuration with VBA (Page 1570)

Average Property

Description

TRUE, when the average value from the last 15 values should be calculated. BOOLEAN write-read access.

Example:

The "BarGraphConfiguration()" procedure configures In this example, value averaging will be activated:

```
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 1819)

Axe Property

Description

Defines or returns the axis for displaying the measured value. Value range from 0 to 2.

Axis	Assigned Value
X	0
Y	1
Z	2

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:

```
Sub HMI3DBarGraphConfiguration()
'VBA390
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Axe = 1
End With
End Sub
```

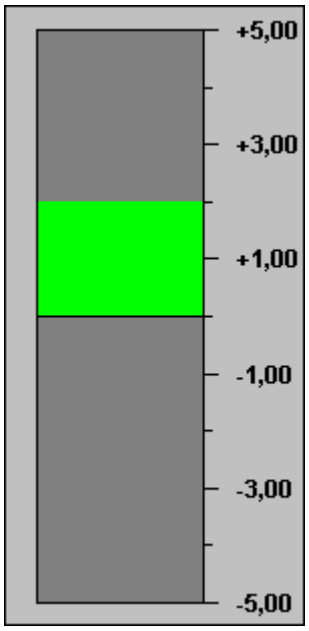
See also

3DBarGraph Object (Page 1805)

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)

Example

The "BarGraphConfiguration()" procedure accesses the properties of the BarGraph object. In this example the axis section will be set to "2".

```
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 1819)

4.4.8.2 B

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".

```
Sub ButtonConfiguration()
'VBA392
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
.BackBorderWidth = 2
```

```
End With
End Sub
```

See also

Slider object (Page 1952)

RoundButton Object (Page 1942)

GroupDisplay Object (Page 1874)

Button Object (Page 1824)

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 1859)
StaticText Object (Page 1956)
Slider object (Page 1952)
TextList Object (Page 1964)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
Polygon Object (Page 1925)
PieSegment Object (Page 1922)
OptionGroup Object (Page 1916)
GroupDisplay Object (Page 1874)
GraphicObject Object (Page 1870)
IOField Object (Page 1886)
Ellipse Object (Page 1853)
Document Object (Page 1847)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)
BarGraph Object (Page 1819)
3DBarGraph Object (Page 1805)

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":

```
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 1819)

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":

```
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 1819)

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 1952)

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":

```
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 1952)

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.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.BackFlashColorOff = RGB(255, 255, 0)
```

```
End With
End Sub
```

See also

BarGraph Object (Page 1819)

StaticText Object (Page 1956)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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":

```
Sub RectangleConfiguration()
'VBA399
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.BackFlashColorOn = RGB(0, 0, 255)
End With
End Suba
```

See also

RoundButton Object (Page 1942)

StaticText Object (Page 1956)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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":

```
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 1805)

BackPictureAlignment property

Description

Round Button

As the "Picture alignment" attribute, defines the position and scaling of the picture placed on the 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-hand side of the round button.
Right justified	The picture is positioned with original proportions, with right justification on the right-hand side of the round button.
Stretched	The picture is scaled to a square and is adapted to the size of the round button.

Background picture

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 larger of the two windows runtime window and process picture. In runtime, it is scaled to the size of the runtime window and is scaled further when opening.

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 1805)

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 1805)

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":

```
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 1805)

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.

```
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 1959)

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 2029)

StatusDisplay Object (Page 1959)

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:

```
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 2027)

StatusDisplay Object (Page 1959)

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 2027)

StatusDisplay Object (Page 1959)

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 1805)

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".

```
Sub HMI3DBarGraphConfiguration()
'VBA409
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.BaseY = 100
End With
End Sub
```

See also

3DBarGraph Object (Page 1805)

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 1822)

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:

```
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 2032)

BitResultInfo Object (Page 1823)

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 1823)

VBA Reference (Page 1664)

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 1823)

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:

```
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 2032)

BitResultInfo Object (Page 1823)

Bold Property

Description

TRUE if the font attribute "Bold" is set for the language-dependent text in the object. BOOLEAN write-read access.

Example:

Note

For this example to work, you must already have configured in the languages concerned.

The following example sets the font attributes of a button for French and English:

```
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
```

See also

Underlined Property (Page 2324)

Size Property (Page 2291)

Parent Property (Page 2241)

LanguageID Property (Page 2158)

Italic Property (Page 2151)

FontFamily Property (Page 2127)

Application Property (Page 2008)

LanguageFont Object (Page 1889)

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 1922)
BorderStyle Property (Page 2043)
TextList Object (Page 1964)
StatusDisplay Object (Page 1959)
StaticText Object (Page 1956)
Slider object (Page 1952)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
PolyLine Object (Page 1928)
OptionGroup Object (Page 1916)
Line Object (Page 1897)
IOField Object (Page 1886)
GraphicObject Object (Page 1870)
EllipseArc Object (Page 1856)
EllipseSegment Object (Page 1859)
Ellipse Object (Page 1853)
CircularArc Object (Page 1831)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)
BarGraph Object (Page 1819)

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":

```
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 1870)

TextList Object (Page 1964)

StatusDisplay Object (Page 1959)

StaticText Object (Page 1956)

Slider object (Page 1952)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

PolyLine Object (Page 1928)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

Line Object (Page 1897)

IOField Object (Page 1886)

EllipseArc Object (Page 1856)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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 1942)

Button Object (Page 1824)

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 1942)

Button Object (Page 1824)

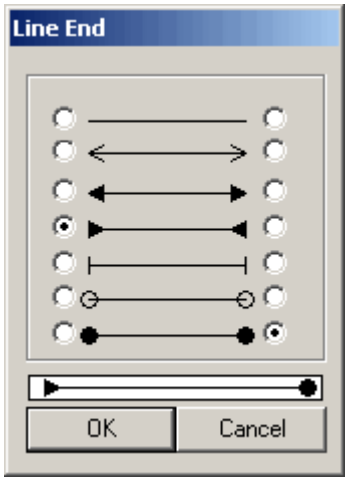
BorderEndStyle Property

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:

```
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 1928)

Line Object (Page 1897)

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":

```
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 1942)

StatusDisplay Object (Page 1959)

StaticText Object (Page 1956)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

Rectangle Object (Page 1939)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

Line Object (Page 1897)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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":

```
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 1956)
StatusDisplay Object (Page 1959)
Slider object (Page 1952)
TextList Object (Page 1964)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
PolyLine Object (Page 1928)
Polygon Object (Page 1925)
PieSegment Object (Page 1922)
OptionGroup Object (Page 1916)
Line Object (Page 1897)
GraphicObject Object (Page 1870)
IOField Object (Page 1886)
EllipseSegment Object (Page 1859)
EllipseArc Object (Page 1856)
Ellipse Object (Page 1853)
CircularArc Object (Page 1831)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)

BorderStyle Property

Description

Defines or returns the line style for the object. Value range from 0 to 4:

Line style	Assigned Value
———	0
— —	1
- - - - - -	2
— · —	3
— · · —	4

Example:

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the line style will be set to "1":

```
Sub RectangleConfiguration()
'VBA422
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.BorderStyle = 1
End With
End Sub
```

See also

IOField Object (Page 1886)

StatusDisplay Object (Page 1959)

StaticText Object (Page 1956)

Slider object (Page 1952)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PolyLine Object (Page 1928)

TextList Object (Page 1964)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

Line Object (Page 1897)

GraphicObject Object (Page 1870)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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.HMIObjects.AddHMIObject("Circle1", "HMICircle")
With objCircle
.BorderWidth = 2
End With
End Sub
```

See also

IOField Object (Page 1886)
StatusDisplay Object (Page 1959)
StaticText Object (Page 1956)
Slider object (Page 1952)
TextList Object (Page 1964)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
PolyLine Object (Page 1928)
Polygon Object (Page 1925)
PieSegment Object (Page 1922)
OptionGroup Object (Page 1916)
Line Object (Page 1897)
GraphicObject Object (Page 1870)
EllipseSegment Object (Page 1859)
EllipseArc Object (Page 1856)
Ellipse Object (Page 1853)
CircularArc Object (Page 1831)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)

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

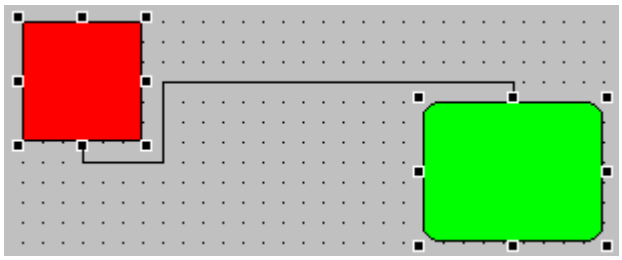
objConnection Object (Page 1912)

BottomConnectedConnectionPointIndex Property

Description

Returns the connection point on the object to which the connector is connected.

Connection Point	Assigned Value
Up	0
Right	1
Down	2
Left	3



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 1912)

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 2048)

OptionGroup Object (Page 1916)

CheckBox Object (Page 1827)

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>":

```
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 2047)

OptionGroup Object (Page 1916)

CheckBox Object (Page 1827)

BoxType Property

Description

Defines or returns the field type. Value range from 0 to 2.

Field type	Assigned Value
Edition	0
Input	1
I/O field	2

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 1886)

Button1Width Property

Description

Defines or returns for the Group Display object the width of button 1 in pixels.

When the SameSize property is set to TRUE, all the buttons are specified the same width.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the width of button 1 will be set to "50.

```
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 2276)

Button4Width Property (Page 2051)

Button3Width Property (Page 2051)

Button2Width Property (Page 2050)

GroupDisplay Object (Page 1874)

Button2Width Property

Description

Defines or returns for the Group Display object the width of button 2 in pixels.

When the SameSize property is set to TRUE, all the buttons are specified the same width.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the width of button 2 will be set to "50.

```
Sub GroupDisplayConfiguration()
'VBA428
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.Button2Width = 50
End With
End Sub
```

See also

SameSize Property (Page 2276)

Button4Width Property (Page 2051)

Button3Width Property (Page 2051)

Button1Width Property (Page 2049)

GroupDisplay Object (Page 1874)

Button3Width Property

Description

Defines or returns for the Group Display object the width of button 3 in pixels.

When the SameSize property is set to TRUE, all the buttons are specified the same width.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the width of button 3 will be set to "50.

```
Sub GroupDisplayConfiguration()
'VBA429
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.Button3Width = 50
End With
End Sub
```

See also

SameSize Property (Page 2276)

Button4Width Property (Page 2051)

Button2Width Property (Page 2050)

Button1Width Property (Page 2049)

GroupDisplay Object (Page 1874)

Button4Width Property

Description

Defines or returns for the Group Display object the width of button 4 in pixels.

When the SameSize property is set to TRUE, all the buttons are specified the same width.

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the width of button 4 will be set to "50.

```
Sub GroupDisplayConfiguration()
'VBA430
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.Button4Width = 50
End With
End Sub
```

See also

Button1Width Property (Page 2049)

SameSize Property (Page 2276)

Button3Width Property (Page 2051)

Button2Width Property (Page 2050)

GroupDisplay Object (Page 1874)

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".

```
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 1952)

4.4.8.3 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 1919)

ApplicationWindow Object (Page 1817)

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 1919)

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
```

See also

TypeAlarmHigh Property (Page 2316)

ColorAlarmHigh Property (Page 2066)

AlarmHigh Property (Page 2002)

BarGraph Object (Page 1819)

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
'Set analysis to absolute
.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 2067)

TypeAlarmLow Property (Page 2317)

AlarmLow Property (Page 2002)

BarGraph Object (Page 1819)

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:

```
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 2222)

Configuring Menus and Toolbars (Page 1572)

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 2318)

LimitHigh4 Property (Page 2193)

ColorLimitHigh4 Property (Page 2069)

BarGraph Object (Page 1819)

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 2070)

TypeLimitHigh5 Property (Page 2318)

LimitHigh4 Property (Page 2193)

BarGraph Object (Page 1819)

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.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

TypeLimitLow4 Property (Page 2319)

LimitLow4 Property (Page 2195)

ColorLimitLow4 Property (Page 2071)

BarGraph Object (Page 1819)

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 2320)

LimitLow5 Property (Page 2196)

ColorLimitLow5 Property (Page 2072)

BarGraph Object (Page 1819)

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 2321)

ToleranceHigh Property (Page 2308)

ColorToleranceHigh Property (Page 2073)

BarGraph Object (Page 1819)

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".

```
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 1819)

TypeToleranceLow Property (Page 2322)

ToleranceLow Property (Page 2308)

ColorToleranceLow Property (Page 2074)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis to absolute
.TypeWarningHigh = False
'Activate monitoring
.CheckWarningHigh = True
'Set barcolor to "red"
.ColorWarningHigh = RGB(255, 0, 0)
'Set upper limit to "75"
.WarningHigh = 75
End With
End Sub
```

See also

WarningHigh Property (Page 2406)

TypeWarningHigh Property (Page 2322)

ColorWarningHigh Property (Page 2076)

BarGraph Object (Page 1819)

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 2407)

TypeWarningLow Property (Page 2323)

ColorWarningLow Property (Page 2077)

BarGraph Object (Page 1819)

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:

```
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 1886)

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:

```
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 1886)

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:

```
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 1919)

ApplicationWindow Object (Page 1817)

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.

Example:

--

See also

GroupDisplay Object (Page 1874)

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.

```
Sub BarGraphLimitConfiguration()
'VBA449
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 "red"
.ColorAlarmHigh = RGB(255, 0, 0)
'Set upper limit to "50"
.AlarmHigh = 50
End With
End Sub
```

See also

CheckAlarmHigh Property (Page 2055)

BarGraph Object (Page 1819)

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 2056)

BarGraph Object (Page 1819)

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 1952)

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 1819)

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 2057)

BarGraph Object (Page 1819)

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".

```
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 2058)

BarGraph Object (Page 1819)

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 2059)

BarGraph Object (Page 1819)

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".

```
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 2060)

BarGraph Object (Page 1819)

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 2061)

BarGraph Object (Page 1819)

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 2061)

BarGraph Object (Page 1819)

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":

```
Sub SliderConfiguration()
'VBA459
```

```
Dim objSlider As HMISlider
Set objSlider = ActiveDocument.HMIObjects.AddHMIObject("SliderObject1", "HMISlider")
With objSlider
.ColorTop = RGB(255, 128, 0)
End With
End Sub
```

See also

Slider object (Page 1952)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
'Set analysis to absolute
.TypeWarningHigh = False
'Activate monitoring
.CheckWarningHigh = True
'Set barcolor to "red"
.ColorWarningHigh = RGB(255, 0, 0)
'Set upper limit to "75"
.WarningHigh = 75
End With
End Sub
```

See also

CheckWarningHigh Property (Page 2062)

BarGraph Object (Page 1819)

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 2063)

BarGraph Object (Page 1819)

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 1847)

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:

```
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.

```
Sub Document_Opened(CancelForwarding As Boolean)
'VBA462
MsgBox Application.Commandline
End Sub
```

See also

Application Object (Page 1814)

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:

```
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 2295)

ScriptInfo Object (Page 1948)

ConfigurationFileName Property

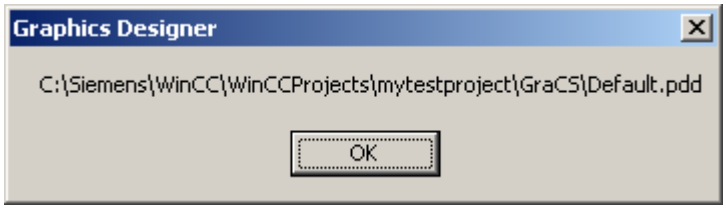
Description

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 2008)

Application Object (Page 1814)

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 HMICircle
Dim objRectangle As HMIRectangle
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, "HMICircle")
Set objRectangle = objDocument.HMIObjects.AddHMIObject("Rectangle" & iIndex,
"HMIRectangle")
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 1990)
Views Object (Listing) (Page 1993)
VariableStateValues Object (Listing) (Page 1987)
ToolbarItems Object (Listing) (Page 1973)
Toolbars Object (Listing) (Page 1968)
SymbolLibraries Object (Listing) (Page 1963)
Selection Object (Listing) (Page 1949)
Properties Object (Listing) (Page 1931)
HMIObjects Object (Listing) (Page 1884)
MenuItems Object (Listing) (Page 1909)
Menus Object (Listing) (Page 1904)
Layers Object (Listing) (Page 1895)
LanguageTexts Object (Listing) (Page 1893)
LanguageFonts Object (Listing) (Page 1890)
GroupedObjects Object (Listing) (Page 1877)
FolderItems Object (Listing) (Page 1868)
Events Object (Listing) (Page 1863)
Documents Object (Listing) (Page 1850)
HMIDefaultObjects Object (Listing) (Page 1878)
DataLanguages Object (Listing) (Page 1842)
ConnectionPoints Object (Listing) (Page 1837)
AnalogResultInfos Object (Listing) (Page 1813)
Actions Object (Listing) (Page 1810)

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:

```
Sub ShowDataLanguage()
```

```
'VBA466
MsgBox Application.CurrentDataLanguage
End Sub
```

See also

Application Property (Page 2008)

DataLanguageChanged Event (Page 1681)

Language-Dependent Configuration with VBA (Page 1570)

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:

```
Sub ShowDesktopLanguage()
'VBA467
MsgBox Application.CurrentDesktopLanguage
End Sub
```

See also

Application Property (Page 2008)

Application Object (Page 1814)

DesktopLanguageChanged event (Page 1682)

Language-Dependent Configuration with VBA (Page 1570)

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.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
Application.ActiveDocument.CursorMode = True
With objIOField
.CursorControl = True
End With
End Sub
```

See also

TabOrderAlpha Property (Page 2301)

TabOrderSwitch Property (Page 2299)

CursorMode Property (Page 2085)

ActiveDocument Property (Page 1996)

TextList Object (Page 1964)

IOField Object (Page 1886)

CursorMode Property

Description

TRUE if the "Alpha Cursor" mode is to be activated. FALSE if the "Tab Order Cursor" 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 2084)

ActiveDocument Property (Page 1996)

Documents Object (Listing) (Page 1850)

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
```

See also

Application Property (Page 2008)

ActiveDocument Property (Page 1996)

Menu Object (Page 1903)

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:

```
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 2008)

ActiveDocument Property (Page 1996)

Toolbar Object (Page 1967)

CycleName Property

Description

Returns the name of the specified tag trigger. Read only access.

Example:

--

See also

VariableTrigger Object (Page 1989)

CycleTime Property

Description

Returns the cycle time of the specified tag trigger. Read only access.

Example:

--

See also

VariableTrigger Object (Page 1989)

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:

```
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 1989)

Configuring Dynamics in the Properties of Pictures and Objects (Page 1632)

4.4.8.4 D

DataFormat Property

Description

Defines or returns the data type of the IOField object. Value range from 0 to 3.

Data type	Assigned Value
Binary	0
Decimal	1
String	2
Hexadecimal	3

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:

```
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 1886)

DefaultHMIObjects Property

Description

Returns the HMIDefaultObjects listing.

Example:

The "ShowDefaultObjectNames()" procedure outputs all the object names contained in the HMIDefaultObjects listing:

```
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 1878)

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:

```
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 2011)

ObjectName Property (Page 2232)

Type Property (Page 2315)

DirectConnection Object (Page 1845)

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.

Position/Bar Axis	Assigned Value
Vertical/Negative	TRUE
Horizontal/Positive	FALSE

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 1952)

3DBarGraph Object (Page 1805)

DisableVBAEvents Property

Description

TRUE if Event Handling is disabled. BOOLEAN write-read access.

Example:

The "DisableVBAEvents()" procedure disables Event Handling:

```
Sub DisableVBAEvents()
'VBA479
Application.DisableVBAEvents = False
End Sub
```

See also

Application Object (Page 1814)

Event Handling (Page 1655)

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:

```
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 &
" / "
Next iIndex1
MsgBox strOutput
End Sub
```

See also

Property Object (Page 1932)

DisplayOptions Property

Description

Defines the assignment of the "Button" or "Round button" object or returns its value. Value range from 0 to 3.

Assignment	Assigned Value
Graphic or text	0
Graphic and text	1
Text only	2
Graphic only	3

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 1824)

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
```

See also

ToolTipText Property (Page 2310)

Label Property (Page 2157)

ToolbarItem Object (Page 1970)

LanguageText Object (Page 1892)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

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 2008)

Application Object (Page 1814)

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:

```
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 1932)

4.4.8.5 E

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:

```
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 1964)

IOField Object (Page 1886)

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:

```
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 1813)

AnalogResultInfo Object (Page 1812)

Add Method (AnalogResultInfos Listing) (Page 1706)

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 HMIMenu
Dim objMenuItem As HMIMenuItem
'
'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 1970)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

Configuring Menus and Toolbars (Page 1572)

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°:

```
Sub PieSegmentConfiguration()
'VBA487
Dim objPieSegment As HMIPieSegment
Set objPieSegment = ActiveDocument.HMIObjects.AddHMIObject("PieSegment1", "HMIPieSegment")
With objPieSegment
.StartAngle = 40
.EndAngle = 180
End With
End Sub
```

See also

StartAngle Property (Page 2296)

PieSegment Object (Page 1922)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

CircularArc Object (Page 1831)

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(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):

Index	EventType (depending upon the object used)
0	hmiEventTypeNotDefined
1	hmiEventTypeMouseClick
2	hmiEventTypeMouseLButtonDown
3	hmiEventTypeMouseLButtonUp
4	hmiEventTypeMouseRButtonDown
5	hmiEventTypeMouseRButtonUp
6	hmiEventTypeKeyboardDown
7	hmiEventTypeKeyboardUp
8	hmiEventTypeObjectChange

Index	EventType (depending upon the object used)
9	hmiEventTypePropertyChange
10	hmiEventTypePropertyStateChange
11	hmiEventTypePictureOpen
12	hmiEventTypePictureClose
13	hmiEventTypeObjectDefined
14	hmiEventTypeFocusEnter
15	hmiEventTypeLastTriggerType
16	hmiEventTypeObjSpecificTriggerStart

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 1863)

Configuring Event-Driven Actions with VBA (Page 1644)

EventType Property

Description

Returns the event type that is configured on the specified object.

Index	EventType (depending upon the object used)
0	hmiEventTypeNotDefined
1	hmiEventTypeMouseClick
2	hmiEventTypeMouseLButtonDown
3	hmiEventTypeMouseLButtonUp
4	hmiEventTypeMouseRButtonDown
5	hmiEventTypeMouseRButtonUp
6	hmiEventTypeKeyboardDown
7	hmiEventTypeKeyboardUp
8	hmiEventTypeObjectChange
9	hmiEventTypePropertyChange
10	hmiEventTypePropertyStateChange
11	hmiEventTypePictureOpen
12	hmiEventTypePictureClose
13	hmiEventTypeObjectDefined
14	hmiEventTypeFocusEnter
15	hmiEventTypeLastTriggerType
16	hmiEventTypeObjSpecificTriggerStart

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":

```
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 1863)

Configuring Event-Driven Actions with VBA (Page 1644)

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:

```
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 1819)

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:

```
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 1952)

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:

```
Sub DocConfiguration()
'VBA815
Dim objDoc As Document
Set objDoc = ActiveDocument
With objDoc
.ExtendedZoomingEnable = True
End With
End Sub
```

4.4.8.6 F

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 AddFaceplateInstance()
'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
```

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".

```
Sub RectangleConfiguration()
'VBA493
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.FillColor = RGB(255, 255, 0)
End With
End Sub
```

See also

Button Object (Page 1824)

StaticText Object (Page 1956)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

GroupDisplay Object (Page 1874)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Document Object (Page 1847)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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()
'VBA494
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.Filling = True
End With
End Sub
```

See also

FillingIndex Property (Page 2107)

StaticText Object (Page 1956)

Slider object (Page 1952)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

GraphicObject Object (Page 1870)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
.Filling = True
.FillingIndex = 50
End With
End Sub
```

See also

PieSegment Object (Page 1922)

FillColor Property (Page 2104)

BackColor Property (Page 2017)

StaticText Object (Page 1956)

Slider object (Page 1952)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

OptionGroup Object (Page 1916)

GraphicObject Object (Page 1870)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Circle Object (Page 1828)

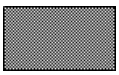
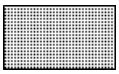
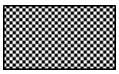
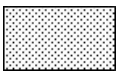
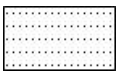
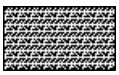
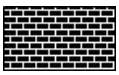
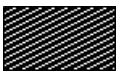
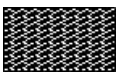
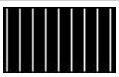
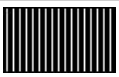
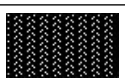
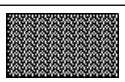
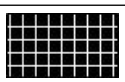
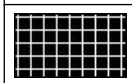
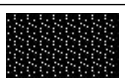
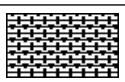
CheckBox Object (Page 1827)

Button Object (Page 1824)

FillStyle Property

Description

Defines or returns the fill style for the object.

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
< Transparent	>65536				
< Solid	>0				
	1048576		196611		196627
	1048577		196612		196628
	1048578		196613		196629
	1048579		196614		196630
	1048832		196615		196631
	1048833		196616		196632
	1048834		196617		196633
	1048835		196618		196634
	131072		196619		196635
	131073		196620		196636
	131074		196621		196637
	131075		196622		196638
	131076		196623		196639
	196608		196624		196640

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
	196609		196625		196641
	196610		196626		196642

Example

The "RectangleConfiguration()" procedure accesses the properties of the rectangle. In this example the fill pattern will be set to the value "196642":

```
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 1964)

StaticText Object (Page 1956)

Slider object (Page 1952)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GraphicObject Object (Page 1870)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Document Object (Page 1847)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

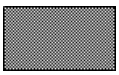
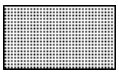
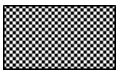
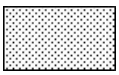
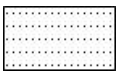
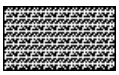
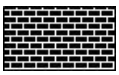
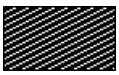
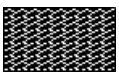
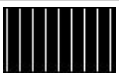
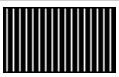
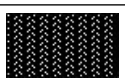
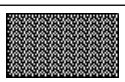
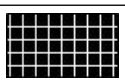
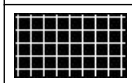
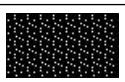
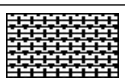
Button Object (Page 1824)

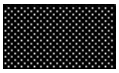
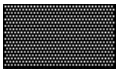
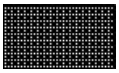
BarGraph Object (Page 1819)

FillStyle2 Property

Description

Defines or returns the fill pattern of the bar for the BarGraph object.

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
Transparent	65536				
Solid	0				
	1048576		196611		196627
	1048577		196612		196628
	1048578		196613		196629
	1048579		196614		196630
	1048832		196615		196631
	1048833		196616		196632
	1048834		196617		196633
	1048835		196618		196634
	131072		196619		196635
	131073		196620		196636
	131074		196621		196637
	131075		196622		196638
	131076		196623		196639
	196608		196624		196640

Fill pattern	Value	Fill pattern	Value	Fill pattern	Value
	196609		196625		196641
	196610		196626		196642

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 1819)

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 1942)

StaticText Object (Page 1956)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

Rectangle Object (Page 1939)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

Ellipse Object (Page 1853)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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:

```
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 1956)

StatusDisplay Object (Page 1959)

Slider object (Page 1952)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

Line Object (Page 1897)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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.AddHMIObject("StatusDisplay1",
"HMIStatusDisplay")
With objsDisplay
.FlashFlashPicture = True
End With
End Sub
```

See also

StatusDisplay Object (Page 1959)

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:

```
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 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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 1959)

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 2117)

StatusDisplay Object (Page 1959)

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:

```
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 2115)

StatusDisplay Object (Page 1959)

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":

```
Sub StatusDisplayConfiguration()
'VBA505
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 2116)

StatusDisplay Object (Page 1959)

FlashRate Property

Description

Defines or returns the flash frequency of the GroupDisplay object. Value range from 0 to 2.

Flash frequency	Assigned Value
Slow (approx. 0.5 Hz)	0
Medium (approx. 2 Hz)	1
Fast (approx. 8 Hz)	2

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 cycle etc.).

Example:

The "GroupDisplayConfiguration()" procedure accesses the properties of the Group Display. In this example the flash frequency will be set to "Medium":

```
Sub GroupDisplayConfiguration()
'VBA506
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.FlashRate = 1
End With
End Sub
```

See also

GroupDisplay Object (Page 1874)

FlashRateBackColor Property

Description

Defines or returns the flash frequency for the object background. Value range from 0 to 2.

Flash frequency	Assigned Value
Slow (approx. 0.5 Hz)	0
Medium (approx. 2 Hz)	1
Fast (approx. 8 Hz)	2

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 cycle etc.).

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 1956)
Slider object (Page 1952)
TextList Object (Page 1964)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
Polygon Object (Page 1925)
PieSegment Object (Page 1922)
OptionGroup Object (Page 1916)
GraphicObject Object (Page 1870)
IOField Object (Page 1886)
EllipseSegment Object (Page 1859)
Ellipse Object (Page 1853)
Circle Object (Page 1828)
CheckBox Object (Page 1827)
Button Object (Page 1824)
BarGraph Object (Page 1819)

FlashRateBorderColor Property

Description

Defines or returns the flash frequency for the lines of the object. Value range from 0 to 2.

Flash frequency	Assigned Value
Slow (approx. 0.5 Hz)	0
Medium (approx. 2 Hz)	1
Fast (approx. 8 Hz)	2

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 cycle etc.).

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the flash frequency for the border will be set to "Medium":

```
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 1952)

StatusDisplay Object (Page 1959)

StaticText Object (Page 1956)

TextList Object (Page 1964)

RoundRectangle Object (Page 1945)

RoundButton Object (Page 1942)

Rectangle Object (Page 1939)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

PieSegment Object (Page 1922)

OptionGroup Object (Page 1916)

Line Object (Page 1897)

GraphicObject Object (Page 1870)

IOField Object (Page 1886)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

CheckBox Object (Page 1827)

Button Object (Page 1824)

FlashRateFlashPic Property

Description

Defines or returns the flash frequency for the status display. Value range from 0 to 2.

Flash frequency	Assigned Value
Slow (approx. 0.5 Hz)	0
Medium (approx. 2 Hz)	1
Fast (approx. 8 Hz)	2

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 cycle etc.).

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":

```
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 1959)

FlashRateForeColor Property

Description

Defines or returns the flash frequency for the object label. Value range from 0 to 2.

Flash frequency	Assigned Value
Slow (approx. 0.5 Hz)	0
Medium (approx. 2 Hz)	1
Fast (approx. 8 Hz)	2

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 cycle etc.).

Example:

The "ButtonConfiguration()" procedure accesses the properties of the button. In this example the flash frequency for the label will be set to "Medium":

```
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 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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:

```
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 symbollibraries:
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 symbollibrary:
iMaxFolder = .FolderItems.Count
MsgBox "Number of FolderItems in " & .Name & " : " & iMaxFolder
'
'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 "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

SymbolLibraries Object (Listing) (Page 1963)

SymbolLibrary Object (Page 1962)

FolderItems Object (Listing) (Page 1868)

FolderItem Object (Page 1867)

Accessing the component library with VBA (Page 1593)

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 symbollibraries:
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 symbollibrary:
iMaxFolder = .FolderItems.Count
MsgBox "Number of FolderItems in " & .Name & " : " & iMaxFolder
'
'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 "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 1867)

SymbolLibraries Object (Listing) (Page 1963)

SymbolLibrary Object (Page 1962)

FolderItems Object (Listing) (Page 1868)

Accessing the component library with VBA (Page 1593)

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":

```
Sub ButtonConfiguration()
'VBA513
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
```

```
With objButton
.FontBold = True
End With
End Sub
```

See also

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

FontFamily Property

Description

Defines or returns the language-dependent font.

Example:

The following example sets the font attributes of a button for French and English:

```
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
With objButton
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 2324)

Size Property (Page 2291)

Parent Property (Page 2241)

Italic Property (Page 2151)

LanguageID Property (Page 2158)

Bold Property (Page 2034)

Application Property (Page 2008)

LanguageFont Object (Page 1889)

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 1956)

TextList Object (Page 1964)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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 1827)

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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
```

See also

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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
```

See also

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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":

```
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 1824)

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

BarGraph Object (Page 1819)

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":

```
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 1827)

TextList Object (Page 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

Button Object (Page 1824)

BarGraph Object (Page 1819)

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 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

GroupDisplay Object (Page 1874)

CheckBox Object (Page 1827)

Button Object (Page 1824)

BarGraph Object (Page 1819)

4.4.8.7 G-H

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

--

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:

```
Sub ActiveDocumentConfiguration()
'VBA521
Application.ActiveDocument.Grid = True
End Sub
```

See also

GridWidth Property (Page 2137)

GridHeight Property (Page 2136)

GridColor Property (Page 2136)

ActiveDocument Property (Page 1996)

Application Property (Page 2008)

Document Object (Page 1847)

Application Object (Page 1814)

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 2135)

ActiveDocument Property (Page 1996)

Application Property (Page 2008)

Document Object (Page 1847)

Application Object (Page 1814)

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":

```
Sub ActiveDocumentConfiguration()
'VBA523
Application.ActiveDocument.Grid = True
Application.ActiveDocument.GridHeight = 8
End Sub
```

See also

GridWidth Property (Page 2137)

Grid Property (Page 2135)

ActiveDocument Property (Page 1996)

Application Property (Page 2008)

Document Object (Page 1847)

Application Object (Page 1814)

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 2135)

GridHeight Property (Page 2136)

ActiveDocument Property (Page 1996)

Application Property (Page 2008)

Document Object (Page 1847)

Application Object (Page 1814)

GroupParent Property

Description

Returns the higher-ranking object in the specified group object. Read-only access.

Example:

--

See also

Group Object (Page 1873)

ActiveDocument Property (Page 1996)

GroupedObjects Object (Listing) (Page 1877)

Document Object (Page 1847)

Application Object (Page 1814)

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:

```
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 1873)

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 1991)

HMIObject Object (Page 1882)

Document Object (Page 1847)

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:

```
Private Sub Document_Opened(CancelForwarding As Boolean)
'VBA802
MsgBox Me.Hide
End Sub
```

See also

Open Method (Page 1780)

Add Method (Documents Listing) (Page 1709)

Document Object (Page 1847)

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:

```
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 1886)

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: :

```
Sub AddCircle()
'VBA529
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("my Circle", "HMICircle")
End Sub
```

See also

Document Object (Page 1847)

Hotkey Property

Description

Defines or returns the function key for a mouse action in the case of the Button object.

Function key	Assigned Value
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123

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.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
.Hotkey = 116
End With
End Sub
```

See also

Button Object (Page 1824)

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:

```
Sub BarGraphConfiguration()
'VBA531
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Hysteresis = True
End With
End Sub
```

See also

BarGraph Object (Page 1819)

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%":

```
Sub BarGraphConfiguration()
'VBA532
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Hysteresis = True
.HysteresisRange = 4
End With
End Sub
```

See also

BarGraph Object (Page 1819)

Hysteresis Property (Page 2143)

4.4.8.8 I - K

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.

```
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 1973)

ToolbarItem Object (Page 1970)

How to Add a New Icon to the Toolbar (Page 1584)

How to Create a New Application-Specific Toolbar (Page 1582)

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.

yes	Size and position of the picture window change with the shift or scaling of the process picture
No	Size and position of the picture window are independent of the process picture and only defined by the "Window mode" attribute

Example

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>":

```
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 1897)

FlashPicture Property (Page 2116)

BasePicture Property (Page 2028)

ActualPointTop Property (Page 1998)

ActualPointLeft Property (Page 1997)

StatusDisplay Object (Page 1959)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

OptionGroup Object (Page 1916)

InputValue property

Description

Defines the value to be specified by the user in the IO field. You can configure the IO field with this attribute so that the input value changed by the user is also displayed in the IO field after confirming with the <Return> key.

This requires that you configure a direct connection between the "Input value" and "Output value" attributes. Without this direct connection, the entered value is deleted after acceptance with the <Return> key. 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
Application.ActiveDocument.Views(1).Activate
'
'Output state of copy:
MsgBox Application.ActiveDocument.Views(1).IsActive
End Sub
```

See also

ActiveDocument Property (Page 1996)

View Object (Page 1991)

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:

```
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 1814)

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 HMIProperty
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 1932)

HMIObject Object (Page 1882)

HMIObjectPropertyChanged Event (Page 1690)

Executing VBA Macros in Graphics Designer (Page 1566)

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 2324)

Size Property (Page 2291)

Parent Property (Page 2241)

LanguageID Property (Page 2158)

FontFamily Property (Page 2127)

Bold Property (Page 2034)

Application Property (Page 2008)

LanguageFont Object (Page 1889)

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:

```
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 1990)

VariableStateValues Object (Listing) (Page 1987)

AnalogResultInfos Object (Listing) (Page 1813)

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 2154)

TextList Object (Page 1964)

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 1964)

ItemBorderStyle Property

Description

Defines or returns the dividing line style in the selection list for the TextList object. Value range from 0 to 4.

Line style	Assigned Value
———	0
— —	1
------	2
— · —	3
—··—	4

Example:

The "TextListConfiguration()" procedure accesses the properties of the object TextList. In this example the dividing line style will be set to "1":

```
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 1964)

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":

```
'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 1964)

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"):

```
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:

```
Sub Document_MenuItemClicked(ByVal MenuItem As IHMIMenuItem)
'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 1970)

MenuItem Object (Page 1906)

InsertToolbarItem Method (Page 1769)

InsertMenuItem Method (Page 1765)

ToolbarItemClicked Event (Page 1701)

MenuItemClicked Event (Page 1695)

Creating Customized Menus and Toolbars (Page 1573)

4.4.8.9 L

La

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 2086)

MenuItems Object (Listing) (Page 1909)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

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 1842)

DataLanguage Object (Page 1841)

LanguageName Property

Description

Returns the project language. STRING read access.

Example:

The "DataLanguages()" procedure outputs the project languages together with their language identifiers:

```
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 & " " & strLangName, vbOKCancel)
If vbCancel = iAnswer Then Exit For
Next objDataLang
End Sub
```

See also

DataLanguages Object (Listing) (Page 1842)

DataLanguage Object (Page 1841)

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:

```
Sub TextListConfiguration()
'VBA551
Dim objTextList As HMITextList
Set objTextList = ActiveDocument.HMIObjects.AddHMIObject("myTextList", "HMITextList")
With objTextList
.LanguageSwitch = True
End With
End Sub
```

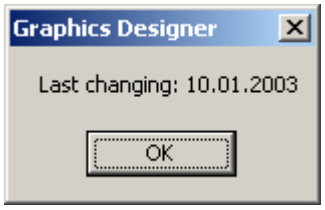
See also

TextList Object (Page 1964)

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:

```
Sub ActiveDocumentConfiguration()
'VBA552
Dim varLastDocChange As Variant
varLastDocChange = Application.ActiveDocument.LastChange
MsgBox "Last changing: " & varLastDocChange
End Sub
```

See also

Document Object (Page 1847)

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 1882)

Editing Layers with VBA (Page 1603)

Layer00Checked Property

Description

TRUE if limit 0 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer00Value and Layer00Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 0 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA554
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer00Checked = True
End With
End Sub
```

See also

Layer00Value Property (Page 2163)

Layer00Color Property (Page 2162)

3DBarGraph Object (Page 1805)

Layer00Color Property

Description

Defines or returns the color for limit 0 of the 3DBarGraph object. 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.

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 will be defined as "Magenta":

```
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

Layer00Checked Property (Page 2161)

3DBarGraph Object (Page 1805)

Layer00Value Property

Description

Defines or returns the value for "limit 0" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer00Checked 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 will be defined as "0":

```
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

Layer00Checked Property (Page 2161)

3DBarGraph Object (Page 1805)

Layer01Checked Property

Description

TRUE if limit 1 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer01Value and Layer01Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 1 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA557
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
```

```
With obj3DBar
.Layer01Checked = True
End With
End Sub
```

See also

Layer01Value Property (Page 2165)

Layer01Color Property (Page 2164)

3DBarGraph Object (Page 1805)

Layer01Color Property

Description

Defines or returns the color for limit 1 of the 3DBarGraph object. 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.

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 1 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA558
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer01Checked = True
.Layer01Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer01Checked Property (Page 2163)

3DBarGraph Object (Page 1805)

Layer01Value Property

Description

Defines or returns the value for "limit 1" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer01Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 1 will be defined as "10":

```
Sub HMI3DBarGraphConfiguration()
'VBA559
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer01Checked = True
.Layer01Value = 10
End With
End Sub
```

See also

Layer01Checked Property (Page 2163)

3DBarGraph Object (Page 1805)

Layer02Checked Property

Description

TRUE if limit 2 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer02Value and Layer02Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 2 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA560
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer02Checked = True
End With
End Sub
```

See also

Layer02Value Property (Page 2167)

Layer02Color Property (Page 2166)

3DBarGraph Object (Page 1805)

Layer02Color Property

Description

Defines or returns the color for limit 2 of the 3DBarGraph object. 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.

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 2 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA561
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
```

```
.Layer02Checked = True
.Layer02Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer02Checked Property (Page 2165)

3DBarGraph Object (Page 1805)

Layer02Value Property

Description

Defines or returns the value for "limit 2" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer02Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 2 will be defined as "20":

```
Sub HMI3DBarGraphConfiguration()
'VBA562
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer02Checked = True
.Layer02Value = 0
End With
End Sub
```

See also

Layer02Checked Property (Page 2165)

3DBarGraph Object (Page 1805)

Layer03Checked Property

Description

TRUE if limit 3 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer03Value and Layer03Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 3 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA563
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer03Checked = True
End With
End Sub
```

See also

Layer03Value Property (Page 2169)

Layer03Color Property (Page 2168)

3DBarGraph Object (Page 1805)

Layer03Color Property

Description

Defines or returns the color for limit 3 of the 3DBarGraph object. 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.

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 3 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
```

```
'VBA564
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer03Checked = True
.Layer03Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer03Checked Property (Page 2167)

3DBarGraph Object (Page 1805)

Layer03Value Property

Description

Defines or returns the value for "limit 3" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer03Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 3 will be defined as "30":

```
Sub HMI3DBarGraphConfiguration()
'VBA565
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer03Checked = True
.Layer03Value = 30
End With
End Sub
```

See also

Layer03Checked Property (Page 2167)

3DBarGraph Object (Page 1805)

Layer04Checked Property

Description

TRUE if limit 4 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer04Value and Layer04Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 4 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA566
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer04Checked = True
End With
End Sub
```

See also

Layer04Value Property (Page 2171)

Layer04Color Property (Page 2170)

3DBarGraph Object (Page 1805)

Layer04Color Property

Description

Defines or returns the color for limit 4 of the 3DBarGraph object. 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.

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 4 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA567
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer04Checked = True
.Layer04Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer04Checked Property (Page 2170)

3DBarGraph Object (Page 1805)

Layer04Value Property

Description

Defines or returns the value for "limit 4" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer00Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 4 will be defined as "40":

```
Sub HMI3DBarGraphConfiguration()
'VBA568
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer04Checked = True
.Layer04Value = 40
End With
End Sub
```

See also

Layer00Checked Property (Page 2161)

3DBarGraph Object (Page 1805)

Layer05Checked Property

Description

TRUE if limit 5 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer05Value and Layer05Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 5 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA569
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer05Checked = True
End With
End Sub
```

See also

Layer05Value Property (Page 2173)

Layer05Color Property (Page 2172)

3DBarGraph Object (Page 1805)

Layer05Color Property

Description

Defines or returns the color for limit 5 of the 3DBarGraph object. 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.

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 5 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA570
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer05Checked = True
.Layer05Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer05Checked Property (Page 2172)

3DBarGraph Object (Page 1805)

Layer05Value Property

Description

Defines or returns the value for "limit 5" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer05Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 5 will be defined as "50":

```
Sub HMI3DBarGraphConfiguration()
'VBA571
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer05Checked = True
.Layer05Value = 50
End With
End Sub
```

See also

Layer05Checked Property (Page 2172)

3DBarGraph Object (Page 1805)

Layer06Checked Property

Description

TRUE if limit 6 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer06Value and Layer06Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 6 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA572
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer06Checked = True
End With
End Sub
```

See also

Layer06Value Property (Page 2175)

Layer06Color Property (Page 2174)

3DBarGraph Object (Page 1805)

Layer06Color Property

Description

Defines or returns the color for limit 6 of the 3DBarGraph object. 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.

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 6 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA573
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer06Checked = True
.Layer06Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer06Checked Property (Page 2174)

3DBarGraph Object (Page 1805)

Layer06Value Property

Description

Defines or returns the value for "limit 6" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer06Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 6 will be defined as "60":

```
Sub HMI3DBarGraphConfiguration()
'VBA574
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer06Checked = True
.Layer06Value = 60
End With
End Sub
```

See also

Layer06Checked Property (Page 2174)

3DBarGraph Object (Page 1805)

Layer07Checked Property

Description

TRUE if limit 7 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer07Value and Layer07Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 7 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA575
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer07Checked = True
End With
End Sub
```

See also

Layer07Value Property (Page 2177)

Layer07Color Property (Page 2176)

3DBarGraph Object (Page 1805)

Layer07Color Property

Description

Defines or returns the color for limit 7 of the 3DBarGraph object. 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.

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 7 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA576
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer07Checked = True
.Layer07Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer07Checked Property (Page 2176)

3DBarGraph Object (Page 1805)

Layer07Value Property

Description

Defines or returns the value for "limit 7" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer07Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 7 will be defined as "70":

```
Sub HMI3DBarGraphConfiguration()
'VBA577
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer07Checked = True
.Layer07Value = 70
End With
End Sub
```

See also

Layer07Checked Property (Page 2176)

3DBarGraph Object (Page 1805)

Layer08Checked Property

Description

TRUE if limit 8 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer08Value and Layer08Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 8 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA578
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer08Checked = True
End With
End Sub
```

See also

Layer08Value Property (Page 2179)

Layer08Color Property (Page 2178)

3DBarGraph Object (Page 1805)

Layer08Color Property

Description

Defines or returns the color for limit 8 of the 3DBarGraph object. 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.

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 8 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA579
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer08Checked = True
.Layer08Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer08Checked Property (Page 2178)

3DBarGraph Object (Page 1805)

Layer08Value Property

Description

Defines or returns the value for "limit 8" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer08Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 8 will be defined as "80":

```
Sub HMI3DBarGraphConfiguration()
'VBA580
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer08Checked = True
.Layer08Value = 80
End With
End Sub
```

See also

Layer08Checked Property (Page 2178)

3DBarGraph Object (Page 1805)

Layer09Checked Property

Description

TRUE if limit 9 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer09Value and Layer09Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 9 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA581
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer09Checked = True
End With
End Sub
```

See also

Layer09Value Property (Page 2181)

Layer09Color Property (Page 2180)

3DBarGraph Object (Page 1805)

Layer09Color Property

Description

Defines or returns the color for limit 9 of the 3DBarGraph object. 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.

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 9 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA582
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer09Checked = True
.Layer09Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer09Checked Property (Page 2180)

3DBarGraph Object (Page 1805)

Layer09Value Property

Description

Defines or returns the value for "limit 9" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer09Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 9 will be defined as "90":

```
Sub HMI3DBarGraphConfiguration()
'VBA583
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer09Checked = True
.Layer09Value = 90
End With
End Sub
```

See also

Layer09Checked Property (Page 2180)

3DBarGraph Object (Page 1805)

Layer10Checked Property

Description

TRUE if limit 10 is to be monitored in the case of the 3DBarGraph object. BOOLEAN write-read access.

Limit value and representation are defined with the Layer10Value and Layer10Color properties.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example, limit 10 will be monitored:

```
Sub HMI3DBarGraphConfiguration()
'VBA584
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer10Checked = True
End With
End Sub
```

See also

Layer10Value Property (Page 2183)

Layer10Color Property (Page 2182)

3DBarGraph Object (Page 1805)

Layer10Color Property

Description

Defines or returns the color for limit 10 of the 3DBarGraph object. 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.

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 10 will be defined as "Magenta":

```
Sub HMI3DBarGraphConfiguration()
'VBA585
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer10Checked = True
.Layer10Color = RGB(255, 0, 255)
End With
End Sub
```

See also

Layer10Checked Property (Page 2182)

3DBarGraph Object (Page 1805)

Layer10Value Property

Description

Defines or returns the value for "limit 10" in the case of the 3DBarGraph object.

Monitoring only takes effect when the Layer10Checked property value is set to TRUE.

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the value for limit 10 will be defined as "100":

```
Sub HMI3DBarGraphConfiguration()
'VBA586
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "HMI3DBarGraph")
With obj3DBar
.Layer10Checked = True
.Layer10Value = 100
End With
End Sub
```

See also

Layer10Checked Property (Page 2182)

3DBarGraph Object (Page 1805)

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 1847)

Editing Layers with VBA (Page 1603)

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 HMIObjects 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 2228)

MinZoom Property (Page 2226)

MaxZoom Property (Page 2208)

Layers Object (Listing) (Page 1895)

Layer Object (Page 1894)

LD - Lo

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 HMILanguageFonts
Dim objLanguageFont As HMILanguageFont
Dim objButton As HMIButton
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 1956)

OptionGroup Object (Page 1916)

LanguageFonts Object (Listing) (Page 1890)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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:

```
Sub CreateMenuItem()
'VBA590
Dim objMenu As HMIMenu
Dim objMenuItem 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:

```
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 1906)

Menu Object (Page 1903)

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 1894)

LanguageTexts Object (Listing) (Page 1893)

FolderItem Object (Page 1867)

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 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 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 HMILanguageTexts
Dim objStatusLngText As HMILanguageText
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 1970)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

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:

```
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 1824)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

CheckBox Object (Page 1827)

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:

```
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:

```
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 1970)

HMIObject Object (Page 1882)

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 1991)

HMIObject Object (Page 1882)

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 1819)

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 1805)

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".

```
Sub BarGraphLimitConfiguration()
'VBA601
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 2318)

CheckLimitHigh4 Property (Page 2057)

BarGraph Object (Page 1819)

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
'Set analysis to absolute
.TypeLimitHigh5 = False
'Activate monitoring
.CheckLimitHigh5 = True
'Set barcolor to "black"
.ColorLimitHigh5 = RGB(0, 0, 0)
'Set upper limit to "80"
.LimitHigh4 = 80
End With
End Sub
```

See also

TypeLimitHigh5 Property (Page 2318)

CheckLimitHigh5 Property (Page 2058)

BarGraph Object (Page 1819)

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 2059)

TypeLimitLow4 Property (Page 2319)

BarGraph Object (Page 1819)

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 1819)

TypeLimitLow5 Property (Page 2320)

CheckLimitLow5 Property (Page 2060)

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":

```
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 2089)

IOField Object (Page 1886)

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":

```
Sub IOFieldConfiguration()
'VBA606
```

```
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.DataFormat = 1
.LimitMin = 0
End With
End Sub
```

See also

DataFormat Property (Page 2089)

IOField Object (Page 1886)

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

ListType Property

Description

Defines or returns the list type in the case of the TextList object. Value range from 0 to 2.

List type	Assigned Value
Decimal	0
Binary	1
Bit	2

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 1964)

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":

```
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 2201)

GroupDisplay Object (Page 1874)

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

Inn 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.

```
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)
Set docOld = Application.Documents.Open(Application.ApplicationDataPath & strFile,
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 1792)

Document Object (Page 1847)

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":

```
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 1874)

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":

```
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 2201)

GroupDisplay Object (Page 1874)

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 2201)

GroupDisplay Object (Page 1874)

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:

```
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 1819)

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:

```
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 1819)

LongStrokesSize Property

Description

The "BarGraphConfiguration()" procedure configures

Example:

In this example the length of the axis section strokes will be set to "10".

```
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 2015)

BarGraph Object (Page 1819)

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:

```
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 1819)

4.4.8.10 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:

```
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 1970)

MenuItem Object (Page 1906)

How to Assign VBA Macros to Menus and Toolbars (Page 1589)

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 1819)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Max = 10
End With
End Sub
```

See also

Slider object (Page 1952)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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:

```
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 1817)

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:

```
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
iAnswer = MsgBox("Layername: " & .Name & vbCrLf & "Min. zoom:  " & .MinZoom & vbCrLf & "Max.
zoom:  " & .MaxZoom, vbOKCancel)
End With
If vbCancel = iAnswer Then Exit For
Next objSingleLayer
End Sub
```

See also

Layer Object (Page 1894)

Editing Layers with VBA (Page 1603)

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":

```
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 1874)

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":

```
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 1874)

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:

```
Sub GroupDisplayConfiguration()
'VBA623
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.MCGUBackFlash = True
End With
End Sub
```

See also

GroupDisplay Object (Page 1874)

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":

```
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 1874)

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":

```
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 1874)

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:

```
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 1874)

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":

```
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 1874)

MCKOBackColorOn 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":

```
Sub GroupDisplayConfiguration()
'VBA628
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
```

```
With objGroupDisplay
.MCKOBackColorOn = RGB(255, 255, 255)
End With
End Sub
```

See also

GroupDisplay Object (Page 1874)

MCKOBackFlash 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:

```
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 1874)

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":

```
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 1874)

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 1874)

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 1874)

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 1874)

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":

```
Sub GroupDisplayConfiguration()
'VBA634
Dim objGroupDisplay As HMIGroupDisplay
Set objGroupDisplay = ActiveDocument.HMIObjects.AddHMIObject("GroupDisplay1",
"HMIGroupDisplay")
With objGroupDisplay
.MCKQBackColorOn = RGB(255, 255, 255)
End With
End Sub
```

See also

GroupDisplay Object (Page 1874)

MCKQBackFlash 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:

```
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 1874)

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":

```
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 1874)

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":

```
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 1874)

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 1874)

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 2224)

GroupDisplay Object (Page 1874)

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:

```
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 1903)

MenuItem Object (Page 1906)

MenuItemType Property

Description

Returns the type for a user-defined menu entry. Read only access.

Returned Value	Type of Menu Entry
0	Separator (Separator)
1	Submenu (SubMenu)
2	Menu Entry (MenuItem)

Example:

The "ShowMenuTypes()" procedure outputs the types for the menu entries in the first user-defined menu:

```
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 1906)

Menu Object (Page 1903)

MessageClass Property

Description

For the GroupDisplay object, defines the appropriate 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.

MessageClass	Assigned Value
AlarmHigh	0
AlarmLow	1
WarningHigh	2
WarningLow	3
Tolerance High	4
Tolerance Low	5
AS Control System Fault	6
AS Process Control Error	7
OS Process Control Error	8

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":

```
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 1874)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Min = 1
End With
End Sub
```

See also

Slider object (Page 1952)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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 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

Layer Object (Page 1894)

Editing Layers with VBA (Page 1603)

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 1948)

Document Object (Page 1847)

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 1919)

ApplicationWindow Object (Page 1817)

4.4.8.11 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 1974)

SymbolLibrary Object (Page 1962)

Property Object (Page 1932)

HMIObject Object (Page 1882)

Layer Object (Page 1894)

FolderItem Object (Page 1867)

Document Object (Page 1847)

Application Object (Page 1814)

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 1867)

Accessing the component library with VBA (Page 1593)

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:

```
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 1664)

PositiveValue Property (Page 2258)

BinaryResultInfo Object (Page 1822)

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:

```
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 & vbCrLf & "Layernumber: " & objLayer.Number &
vbCrLf & "Layersindex: " & iIndex, vbOKCancel)
```

```
iIndex = iIndex + 1
If vbCancel = iAnswer Then Exit For
Next objLayer
End Sub
```

See also

Layer Object (Page 1894)

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 filed 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:

```
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 1964)

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)

Type Property	AutomationName Property	ObjectName Property
hmiSourceTypeConstant	--	Name of the constant (e.g. the picture name)
hmiSourceTypeProperty	Property of the source object (e.g. "Top")	Name of the source object (e.g. "Rectangle_A")
hmiSourceTypePropertyOfThisObject	--	--
hmiSourceTypeVariableDirect	--	Tag name
hmiSourceTypeVariableIndirect	--	Tag name

Destination object type (DestinationLink Property)

Type Property	AutomationName Property	ObjectName Property
hmiDestTypeProperty	Property of the destination object (e.g. "Left")	Name of the destination object (e.g. "Rectangle_A")
hmiDestTypePropertyOfThisObject	--	--
hmiDestTypePropertyOfActualWindow	Property of the destination object (e.g. "Left")	--
hmiDestTypeVariableDirect	--	Tag name
hmiDestTypeVariableIndirect	--	Tag name
hmiDestTypeDirectMessage	--	Tag name
hmiDestTypeIndirectMessage	--	Tag name

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 2315)

SourceLink Property (Page 2293)

DestinationLink Property (Page 2090)

AutomationName Property (Page 2011)

SourceLink Object (Page 1955)

DestLink Object (Page 1843)

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:

```
Sub ConfigureSettingsOfLayer()
'VBA651
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 1847)

Editing Layers with VBA (Page 1603)

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 1919)

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

```
Sub PictureWindowConfig()
'VBA653
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 1919)

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:

```
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 1817)

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 1882)

Document Object (Page 1847)

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()
'VBA656
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 2238)

TextList Object (Page 1964)

Slider object (Page 1952)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

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:

```
Sub IOFieldConfiguration()
'VBA657
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.OperationReport = True
.OperationMessage = True
End With
End Sub
```

See also

OperationMessage Property (Page 2238)

TextList Object (Page 1964)

Slider object (Page 1952)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

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:

```
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 1964)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

IOField Object (Page 1886)

CheckBox Object (Page 1827)

Button Object (Page 1824)

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 2089)

IOField Object (Page 1886)

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 will be set to "00":

```
Sub IOFieldConfiguration()
'VBA660
Dim objIOField As HMIIOField
Set objIOField = ActiveDocument.HMIObjects.AddHMIObject("IOField1", "HMIIOField")
With objIOField
.OutputValue = "00"
End With
End Sub
```

See also

TextList Object (Page 1964)

IOField Object (Page 1886)

4.4.8.12 P-Q

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 1968)
Menu Object (Page 1903)
Document Object (Page 1847)
Views Object (Listing) (Page 1993)
View Object (Page 1991)
VariableTriggers Object (Listing) (Page 1990)
VariableTrigger Object (Page 1989)
VariableStateValues Object (Listing) (Page 1987)
VariableStateValue Object (Page 1986)
Trigger Object (Page 1974)
ToolbarItems Object (Listing) (Page 1973)
ToolbarItem Object (Page 1970)
Toolbar Object (Page 1967)
TextList Object (Page 1964)
SymbolLibraries Object (Listing) (Page 1963)
SymbolLibrary Object (Page 1962)
StatusDisplay Object (Page 1959)
StaticText Object (Page 1956)
SourceLink Object (Page 1955)
Slider object (Page 1952)
Selection Object (Listing) (Page 1949)
ScriptInfo Object (Page 1948)
RoundRectangle Object (Page 1945)
RoundButton Object (Page 1942)
Rectangle Object (Page 1939)
Properties Object (Listing) (Page 1931)
Property Object (Page 1932)
PolyLine Object (Page 1928)
Polygon Object (Page 1925)
PictureWindow Object (Page 1919)
PieSegment Object (Page 1922)
OptionGroup Object (Page 1916)
OLEObject Object (Page 1914)
MenuItems Object (Listing) (Page 1909)
MenuItem Object (Page 1906)
Menus Object (Listing) (Page 1904)
Line Object (Page 1897)
Layers Object (Listing) (Page 1895)
Layer Object (Page 1894)
LanguageTexts Object (Listing) (Page 1893)

PasswordLevel Property

Description

Defines the authorization for operation (e.g. no input or no triggering actions) of the object.

PasswordLevel	Assigned Value
<No Access Security>	0
User Administration	1
Value input	2
Process controlling	3
Picture editing	4
Screen change	5
Window selection	6
Hard copy	7
Confirm alarms	8
Lock alarms	9
Free alarms	10
Message editing	11
Start archive	12
Stop archive	13
Edit archive values	14
Archive editing	15
Action editing	16
Project Manager	17
Activate remote	1000
Configure remote	1001
Just monitor	1002

You must first define the operator authorizations in the User Administrator.

Example:

--

See also

HMIObject Object (Page 1882)

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:

```
Sub ShowDocumentPath()
'VBA663
MsgBox ActiveDocument.Path
End Sub
```

See also

Document Object (Page 1847)

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 1867)

Accessing the component library with VBA (Page 1593)

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 1942)

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.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
.PicDeactTransparent = RGB(255, 0, 0)
```

```
.PicDeactUseTransColor = True
End With
End Sub
```

See also

PicDeactUseTransColor Property (Page 2247)

RoundButton Object (Page 1942)

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 2246)

RoundButton Object (Page 1942)

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 1942)

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 2249)

RoundButton Object (Page 1942)

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
```

See also

PicDownTransparent Property (Page 2248)

RoundButton Object (Page 1942)

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 1870)

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 1870)

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
'
'Toi 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 1942)

PicReferenced Property (Page 2249)

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.HMIObjects.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 1942)

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:

```
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 1919)

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:

```
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 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 1942)

Button Object (Page 1824)

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:

```
Sub RoundButtonConfiguration()
'VBA677
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
.PicUpReferenced = False
End With
End Sub
```

See also

RoundButton Object (Page 1942)

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.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
.PicUpTransparent = RGB(0, 0, 255)
.PicUpUseTransColor = True
End With
End Sub
```

See also

PicUpUseTransColor Property (Page 2255)

RoundButton Object (Page 1942)

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 2254)

RoundButton Object (Page 1942)

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.HMIObjects.AddHMIObject("GraphicObject1",
"HMIGraphicObject")
With objGraphicObject
.PicTransColor = RGB(0, 0, 255)
.PicUseTransColor = True
End With
End Sub
```

See also

PicTransColor Property (Page 2250)

GraphicObject Object (Page 1870)

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:

```
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
MsgBox iIndex & ". corner:" & vbCrLf & "x-coordinate: " & iPosX & vbCrLf & "y-coordinate:
" & iPosY
End With
Next iIndex
End Sub
```

List of links

See also

Index Property (Page 2145)

ActualPointTop Property (Page 1998)

ActualPointLeft Property (Page 1997)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

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 1970)

MenuItem Object (Page 1906)

Menu Object (Page 1903)

InsertSubmenu Method (Page 1767)

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 2229)

BinaryResultInfo Object (Page 1822)

VBA Reference (Page 1664)

PredefinedAngels Property

Description

Defines or returns the depth of the display of the 3DBarGraph object. Value range from 0 to 3.

Display	Assigned Value
Cavalier	0
Isometric	1
Axonometric	2
Freely Defined	3

Example:

The "HMI3DBarGraphConfiguration()" procedure accesses the properties of the 3DBarGraph object. In this example the depth display will be set to "Isometric":

```
Sub HMI3DBarGraphConfiguration()
'VBA685
Dim obj3DBar As HMI3DBarGraph
Set obj3DBar = ActiveDocument.HMIObjects.AddHMIObject("3DBar1", "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 1805)

Pressed Property

Description

TRUE, when the 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":

```
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 1942)

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 1952)

OptionGroup Object (Page 1916)

CheckBox Object (Page 1827)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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:

```
Sub ShowProfileName()
'VBA688
MsgBox Application.ProfileName
End Sub
```

See also

Application Object (Page 1814)

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:

```
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 1811)

AddActiveXControl Method (Page 1714)

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 1814)

ProjectType Property

Description

Returns the project type. Value range from 0 to 2. Read access.

Project type	Assigned Value
Single-user project	0
Multi-user project	1
client project	2

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 1814)

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 1882)

Customized Objects (Page 1626)

Group Objects (Page 1618)

Editing Objects with VBA (Page 1606)

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:

```
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 1948)

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:

```
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 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

DynamicDialog Object (Page 1851)

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_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

DynamicDialog Object (Page 1851)

QualityCodeStateValues Object (Listing) (Page 1936)

4.4.8.13 R

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 1942)

PieSegment Object (Page 1922)

CircularArc Object (Page 1831)

Circle Object (Page 1828)

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":

```
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 2269)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

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":

```
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 2268)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

Ellipse Object (Page 1853)

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 2331)

AnalogResultInfos Object (Listing) (Page 1813)

AnalogResultInfo Object (Page 1812)

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:

```
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 2274)

ReferenceRotationTop Property (Page 2271)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

Line Object (Page 1897)

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:

```
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 2274)

ReferenceRotationLeft Property (Page 2270)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

Line Object (Page 1897)

Relevant Property

Description

TRUE, when the object will be 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:

```
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 1873)

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:

```
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 1851)

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 1819)

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:

0	The standard position of the T-piece is the shape of the letter "T"
90	The "leg" of the "T" points towards the left
180	The "leg" of the "T" points upwards
270	The "leg" of the "T" points to the right

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:

```
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 2271)

ReferenceRotationLeft Property (Page 2270)

PolyLine Object (Page 1928)

Polygon Object (Page 1925)

Line Object (Page 1897)

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()
'VBA702
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 2275)

RoundRectangle Object (Page 1945)

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()
'VBA703
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 2275)

RoundRectangle Object (Page 1945)

4.4.8.14 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 1874)

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":

```
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 2278)

BarGraph Object (Page 1819)

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".

```
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 1819)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Scaling = True
.ScaleColor = RGB(255, 0, 0)
End With
End Sub
```

See also

BarGraph Object (Page 1819)

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.

Bar Scaling	Assigned Value
Linear	0
Logarithmic	1
Automatic	2

Example:

The "BarGraphConfiguration()" procedure configures In this example the bar scaling will be set to "Linear":

```
Sub BarGraphConfiguration()
'VBA708
Dim objBarGraph As HMIBarGraph
```

```
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.ScalingType = 0
.Scaling = True
End With
End Sub
```

See also

Scaling Property (Page 2278)

BarGraph Object (Page 1819)

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:

```
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.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)
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 1948)

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:

```
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 1919)

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 1919)

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.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 1919)

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 2284)

View Object (Page 1991)

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 2283)

View Object (Page 1991)

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":

```
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 1964)

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:

```
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 1882)

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:

```
Sub SelectAllObjectsInActiveDocument()
'VBA715
ActiveDocument.Selection.SelectAll
End Sub
```

See also

Selection Object (Listing) (Page 1949)

Document Object (Page 1847)

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":

```
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 1964)

ServerName Property

Description

Returns the name of the specified ActiveX Control. 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:

```
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 1811)

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:

```
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 1919)

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 1572)

ToolbarItem Object (Page 1970)

MenuItem Object (Page 1906)

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 1874)

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:

```
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 2324)

Parent Property (Page 2241)

LanguageID Property (Page 2158)

Italic Property (Page 2151)

FontFamily Property (Page 2127)

Bold Property (Page 2034)

Application Property (Page 2008)

LanguageFont Object (Page 1889)

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:

```
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 1919)

ApplicationWindow Object (Page 1817)

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":

```
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 1952)

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:

```
Sub ActivateSnapToGrid()
'VBA724
ActiveDocument.SnapToGrid = True
End Sub
```

See also

Document Object (Page 1847)

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 2315)

ObjectName Property (Page 2232)

AutomationName Property (Page 2011)

SourceLink Object (Page 1955)

DirectConnection Object (Page 1845)

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:

```
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 2080)

ScriptInfo Object (Page 1948)

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 2098)

PieSegment Object (Page 1922)

EllipseSegment Object (Page 1859)

EllipseArc Object (Page 1856)

CircularArc Object (Page 1831)

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 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. A status bar entry will be defined for each menu entry:

```
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 1970)

MenuItem Object (Page 1906)

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 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()
'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 1906)

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 1814)

4.4.8.15 T

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 1882)

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

TabOrderOtherAction Property (Page 2303)

TabOrderMouse Property (Page 2302)

TabOrderKeyboard Property (Page 2301)

Document Object (Page 1847)

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:

```
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 1847)

TabOrderKeyboard 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 2303)

TabOrderMouse Property (Page 2302)

TabOrderAllHMIObjects Property (Page 2300)

Document Object (Page 1847)

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 2303)

TabOrderKeyboard Property (Page 2301)

TabOrderAllHMIObjects Property (Page 2300)

Document Object (Page 1847)

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 2302)

TabOrderKeyboard Property (Page 2301)

TabOrderAllHMIObjects Property (Page 2300)

Document Object (Page 1847)

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:

```
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 1970)

MenuItem Object (Page 1906)

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 1919)

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.

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:

```
Sub ButtonConfiguration()
'VBA740
Dim objButton As HMIButton
Set objButton = ActiveDocument.HMIObjects.AddHMIObject("Button1", "HMIButton")
With objButton
.Text = "Button1"
End With
End Sub
```

See also

Button Object (Page 1824)

StaticText Object (Page 1956)

OptionGroup Object (Page 1916)

CheckBox Object (Page 1827)

Toggle Property

Description

TRUE, when the 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:

```
Sub RoundButtonConfiguration()
'VBA741
Dim objRoundButton As HMIRoundButton
Set objRoundButton = ActiveDocument.HMIObjects.AddHMIObject("RButton1", "HMIRoundButton")
With objRoundButton
.Toggle = True
End With
End Sub
```

See also

RoundButton Object (Page 1942)

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:

```
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 2321)

CheckToleranceHigh Property (Page 2061)

BarGraph Object (Page 1819)

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".

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 2322)

CheckToleranceLow Property (Page 2061)

BarGraph Object (Page 1819)

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 1970)

Toolbar Object (Page 1967)

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:

```
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
```

Note

If you want to make the "ToolTipText" property dynamic with the dynamic dialog, you must initialize before the dynamization.

Example:

```
Sub Dyn()
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 1970)

HMIObject Object (Page 1882)

How to Dynamize a Property with Dynamic Dialog (Page 1636)

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:

```
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 1991)

HMIObject Object (Page 1882)

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 BottomConnectedObjectName property can be found in this documentation under the heading "ObjConnection Object".

See also

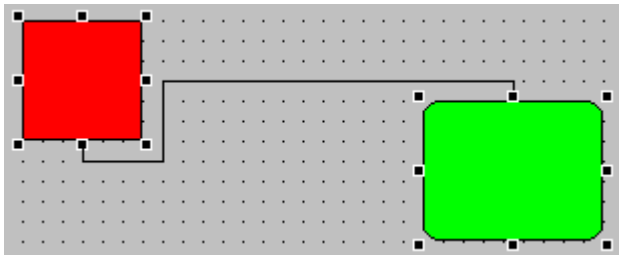
objConnection Object (Page 1912)

TopConnectedConnectionPointIndex Property

Description

Returns the connection point on the object to which the connector is connected.

Connection Point	Assigned Value
Up	0
Right	1
Down	2
Left	3



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 1912)

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 objects other objects shine through. A 100% transparent object is invisible. An invisible object can also be controlled in Runtime.

Example

```
Sub addTransparentObject()
    'VBA849
    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()
'VBA747
Dim objBarGraph As HMIBarGraph
Set objBarGraph = ActiveDocument.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.trend = True
End With
End Sub
```

See also

BarGraph Object (Page 1819)

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":

```
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 2313)

BarGraph Object (Page 1819)

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):

```
Sub AddDynamicAsCSkriptToProperty()
'VBA749
Dim objVBScript As HMIScriptInfo
Dim objCircle As HMICircle

Set objCircle = ActiveDocument.HMIObjects.AddHMIObject("myCircle", "HMICircle")
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 1974)

ScriptInfo Object (Page 1948)

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:

```
Sub RectangleConfiguration()
'VBA750
Dim objRectangle As HMIRectangle
Set objRectangle = ActiveDocument.HMIObjects.AddHMIObject("Rectangle1", "HMIRectangle")
With objRectangle
MsgBox "Objecttype: " & .Type
End With
End Sub
```

See also

Trigger Object (Page 1974)

ToolbarItem Object (Page 1970)

SourceLink Object (Page 1955)

Property Object (Page 1932)

HMIObject Object (Page 1882)

FolderItem Object (Page 1867)

DestLink Object (Page 1843)

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".

```
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 2066)

CheckAlarmHigh Property (Page 2055)

AlarmHigh Property (Page 2002)

BarGraph Object (Page 1819)

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 2067)

CheckAlarmLow Property (Page 2056)

AlarmLow Property (Page 2002)

BarGraph Object (Page 1819)

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.HMIObjects.AddHMIObject("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 2193)

CheckLimitHigh4 Property (Page 2057)

BarGraph Object (Page 1819)

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.HMIObjects.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 2194)

CheckLimitHigh5 Property (Page 2058)

BarGraph Object (Page 1819)

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".

```
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 2195)

ColorLimitLow4 Property (Page 2071)

CheckLimitLow4 Property (Page 2059)

BarGraph Object (Page 1819)

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".

```
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 2196)

ColorLimitLow5 Property (Page 2072)

CheckLimitLow5 Property (Page 2060)

BarGraph Object (Page 1819)

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:

```
Sub BarGraphLimitConfiguration()
'VBA757
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 = "40"
.ToleranceHigh = 40
End With
End Sub
```

See also

ColorToleranceHigh Property (Page 2073)

CheckToleranceHigh Property (Page 2061)

BarGraph Object (Page 1819)

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:

```
Sub BarGraphLimitConfiguration()
'VBA758
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 = "10"
.ToleranceLow = 10
End With
End Sub
```

See also

ToleranceLow Property (Page 2308)

ColorToleranceLow Property (Page 2074)

CheckToleranceLow Property (Page 2061)

BarGraph Object (Page 1819)

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".

```
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 2406)

ColorWarningHigh Property (Page 2076)

CheckWarningHigh Property (Page 2062)

BarGraph Object (Page 1819)

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 2407)

ColorWarningLow Property (Page 2077)

CheckWarningLow Property (Page 2063)

BarGraph Object (Page 1819)

4.4.8.16 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()
'VBA761
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 2291)

Parent Property (Page 2241)

LanguageID Property (Page 2158)

Italic Property (Page 2151)

FontFamily Property (Page 2127)

Bold Property (Page 2034)

Application Property (Page 2008)

LanguageFont Object (Page 1889)

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 1964)

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:

```
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 1964)

UpdateCycle Property

Description

Defines or returns the type and frequency of updates to the picture window in Runtime.

Update Cycle	Assigned Value
Upon change	0
250 ms	1
500 ms	2
1 s	3
2 s	4
5 s	5
10 s	6
1 min	7
5 min	8
10 min	9
1 h	10
User cycle 1	11
User cycle 2	12
User cycle 3	13
User cycle 4	14
User cycle 5	15
Picture cycle	255

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:

```
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 1919)

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:

```
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 2330)

UserValue3 Property (Page 2329)

UserValue2-Eigenschaft (Page 2328)

GroupDisplay Object (Page 1874)

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:

```
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 2330)

UserValue3 Property (Page 2329)

UserValue1 Property (Page 2328)

GroupDisplay Object (Page 1874)

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:

```
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 2330)

UserValue2-Eigenschaft (Page 2328)

UserValue1 Property (Page 2328)

GroupDisplay Object (Page 1874)

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:

```
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 2329)

UserValue2-Eigenschaft (Page 2328)

UserValue1 Property (Page 2328)

GroupDisplay Object (Page 1874)

4.4.8.17 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:

```
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:
objActiveXControl.Properties("BackColor").value = RGB(255, 0, 0)
End Sub
```

See also

Property Object (Page 1932)

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:

```
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 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_LIMIT Property (Page 2365)

VariableStateChecked Property (Page 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VariableStateValue Object (Page 1986)

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 be 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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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:

```
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
'
'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_BAD_DEVICE Property (Page 2340)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
QualityCodeStateValue Object (Page 1934)

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", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
.ResultType = hmiResultTypeAnalog
.AnalogResultInfos.ElseCase = 200
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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

QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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 2389)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_ENGVHIGHLIM Property (Page 2378)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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 2381)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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 2381)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_HIGHLIMITED Property (Page 2359)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_BAD_CONFERROR Property (Page 2338)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_LUV Property (Page 2385)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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 2393)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
Sub AddDynamicDialogToCircleRadiusTypeAnalog()
'VBA773
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 2400)
VALUE_TIMEOUT Property (Page 2376)
VALUE_STARTUP_VALUE Property (Page 2375)
VALUE_SERVERDOWN Property (Page 2373)
VALUE_NOT_ESTABLISHED Property (Page 2372)
VALUE_MIN_RANGE Property (Page 2370)
VALUE_MIN_LIMIT Property (Page 2369)
VALUE_MAX_RANGE Property (Page 2367)
VALUE_MAX_LIMIT Property (Page 2365)
VALUE_INVALID_KEY Property (Page 2361)
VALUE_HARDWARE_ERROR Property (Page 2358)
VALUE_CONVERSION_ERROR Property (Page 2354)
VALUE_ADDRESS_ERROR Property (Page 2333)
VALUE_ACCESS_FAULT Property (Page 2331)
VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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 2367)

VariableStateChecked Property (Page 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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:

```
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
.QualityStateChecked = 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

QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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 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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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.HMIObjects.AddHMIObject("Circle_A", "HMICircle")
Set objDynDialog = objCircle.Radius.CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'NewDynamic1'")
With objDynDialog
.ResultType = hmiResultTypeAnalog
```

```
.AnalogResultInfos.ElseCase = 200
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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_BAD_PROCRELSUB Property (Page 2352)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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 be 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
'
'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_MIN_LIMIT Property (Page 2369)

VariableStateChecked Property (Page 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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
'
'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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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
'
'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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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 2400)
VALUE_TIMEOUT Property (Page 2376)
VALUE_STARTUP_VALUE Property (Page 2375)
VALUE_SERVERDOWN Property (Page 2373)
VALUE_NOT_ESTABLISHED Property (Page 2372)
VALUE_MIN_LIMIT Property (Page 2369)
VALUE_MAX_RANGE Property (Page 2367)
VALUE_MAX_LIMIT Property (Page 2365)
VALUE_INVALID_KEY Property (Page 2361)
VALUE_HARDWARE_ERROR Property (Page 2358)
VALUE_HANDSHAKE_ERROR Property (Page 2356)
VALUE_CONVERSION_ERROR Property (Page 2354)
VALUE_ADDRESS_ERROR Property (Page 2333)
VALUE_ACCESS_FAULT Property (Page 2331)
VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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
'
'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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_STARTUP_VALUE Property (Page 2375)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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
'
'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 2400)

VALUE_TIMEOUT Property (Page 2376)

VALUE_SERVERDOWN Property (Page 2373)

VALUE_NOT_ESTABLISHED Property (Page 2372)

VALUE_MIN_RANGE Property (Page 2370)

VALUE_MIN_LIMIT Property (Page 2369)

VALUE_MAX_RANGE Property (Page 2367)

VALUE_MAX_LIMIT Property (Page 2365)

VALUE_INVALID_KEY Property (Page 2361)

VALUE_HARDWARE_ERROR Property (Page 2358)

VALUE_HANDSHAKE_ERROR Property (Page 2356)

VALUE_CONVERSION_ERROR Property (Page 2354)

VALUE_ADDRESS_ERROR Property (Page 2333)

VALUE_ACCESS_FAULT Property (Page 2331)

VariableStateValue Object (Page 1986)

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 be evaluating the status of a tag. If the tag does not return a status, a substitute value (ElseCase property) is defined:

```
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
'
'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 2400)
VALUE_STARTUP_VALUE Property (Page 2375)
VALUE_SERVERDOWN Property (Page 2373)
VALUE_NOT_ESTABLISHED Property (Page 2372)
VALUE_MIN_RANGE Property (Page 2370)
VALUE_MIN_LIMIT Property (Page 2369)
VALUE_MAX_RANGE Property (Page 2367)
VALUE_MAX_LIMIT Property (Page 2365)
VALUE_INVALID_KEY Property (Page 2361)
VALUE_HARDWARE_ERROR Property (Page 2358)
VALUE_HANDSHAKE_ERROR Property (Page 2356)
VALUE_CONVERSION_ERROR Property (Page 2354)
VALUE_ADDRESS_ERROR Property (Page 2333)
VALUE_ACCESS_FAULT Property (Page 2331)
VariableStateValue Object (Page 1986)

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
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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_BAD_PROCRELSUB Property (Page 2352)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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 2393)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_SIMVAL Property (Page 2395)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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 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:

```
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
.QualityStateChecked = 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_LUV Property (Page 2385)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_HIGHLIMITED Property (Page 2359)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_NONSPECIFIC Property (Page 2391)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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
'
'Activate analysis of qualitycodestate
.QualityStateChecked = 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_LOWLIMITED Property (Page 2363)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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 2387)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_BAD_COMMNUV Property (Page 2336)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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_BAD_PROCRELNOM Property (Page 2350)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SUBSTSET Property (Page 2397)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MISCSTATES Property (Page 2389)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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:

```
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
.QualityStateChecked = 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 2389)
QualityCodeStateChecked Properties (Page 2265)
VALUE_UNCERT_SIMVAL Property (Page 2395)
VALUE_UNCERT_PROCRELNOM Property (Page 2393)
VALUE_UNCERT_NONSPECIFIC Property (Page 2391)
VALUE_UNCERT_MAINTDEM Property (Page 2387)
VALUE_UNCERT_LUV Property (Page 2385)
VALUE_UNCERT_INITVAL Property (Page 2383)
VALUE_UNCERT_ENGVONLIM Property (Page 2381)
VALUE_UNCERT_ENGVLOWLIM Property (Page 2379)
VALUE_UNCERT_ENGVHIGHLIM Property (Page 2378)
VALUE_LOWLIMITED Property (Page 2363)
VALUE_HIGHLIMITED Property (Page 2359)
VALUE_BAD_PROCRELSUB Property (Page 2352)
VALUE_BAD_PROCRELNOM Property (Page 2350)
VALUE_BAD_OUTOFSERV Property (Page 2348)
VALUE_BAD_NOTCONNECTED Property (Page 2346)
VALUE_BAD_NONSPECIFIC Property (Page 2344)
VALUE_BAD_MISCSTATES Property (Page 2342)
VALUE_BAD_DEVICE Property (Page 2340)
VALUE_BAD_CONFERROR Property (Page 2338)
VALUE_BAD_COMMNUV Property (Page 2336)
VALUE_BAD_COMMLUV Property (Page 2335)
QualityCodeStateValue Object (Page 1934)

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 1851)

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 be 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()
'VBA785
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

DynamicDialog Object (Page 1851)

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.

Index	VariableStateType
0	hmiNoVariableState
1	hmiVariableQCState
2	hmiVariableState

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:

```
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 1851)

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 be 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
'
'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

VariableStateValues Object (Listing) (Page 1987)

DynamicDialog Object (Page 1851)

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:

```
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 1990)

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:

```
Sub GetVarName()
'VBA788
Dim objVBScript As HMIScriptInfo
Dim objCircle As HMICircle
Set objCircle = ActiveDocument.HMIObjects.Item("Circle_VariableTrigger")
Set objVBScript = objCircle.Radius.Dynamic
With objVBScript
'Reading out of variablename
MsgBox "The radius is dynamicabled with: " & .Trigger.VariableTriggers.Item(1).VarName
End With
End Sub
```

See also

VariableStateValue Object (Page 1986)

VBAVersion Property

Description

Returns the VBA version number. Read only access.

Example:

In the following example the current VBA version number is output:

```
Sub ShowVBAVersion()
'VBA789
MsgBox Application.VBAVersion
End Sub
```

See also

Application Object (Page 1814)

VBE Property

Description

Returns the VB Extensibility object. Read access.

Example:

--

See also

Application Object (Page 1814)

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:

```
Sub ShowVersionOfGraphicsDesigner()
'VBA791
MsgBox Application.Version
End Sub
```

See also

Application Object (Page 1814)

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 1993)

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

ToolbarItem Object (Page 1970)

MenuItem Object (Page 1906)

HMIObject Object (Page 1882)

Document Object (Page 1847)

Toolbar Object (Page 1967)

Menu Object (Page 1903)

Application Object (Page 1814)

4.4.8.18 W - Z

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".

```
Sub BarGraphLimitConfiguration()
'VBA794
```

```
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

TypeWarningHigh Property (Page 2322)

ColorWarningHigh Property (Page 2076)

CheckWarningHigh Property (Page 2062)

BarGraph Object (Page 1819)

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".

```
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 2323)

ColorWarningLow Property (Page 2077)

CheckWarningLow Property (Page 2063)

BarGraph Object (Page 1819)

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 1882)

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.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 1919)

ApplicationWindow Object (Page 1817)

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.

WindowState	Assigned Value
Maximized	0
Minimized	1
Custom sized	2

Example:

In the following example the window status of the Graphics Designer is output:

```
Sub ShowWindowState()
'VBA798
Dim strState As String
Select Case Application.WindowState
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 1814)

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.HMIObjects.AddHMIObject("Bar1", "HMIBarGraph")
With objBarGraph
.Scaling = True
.ScalingType = 2
.ZeroPoint = 50
.ZeroPointValue = 0
End With
End Sub
```

See also

ZeroPointValue Property (Page 2413)

ScalingType Property (Page 2279)

Scaling Property (Page 2278)

BarGraph Object (Page 1819)

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".

```
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 2412)

ScalingType Property (Page 2279)

Scaling Property (Page 2278)

BarGraph Object (Page 1819)

3DBarGraph Object (Page 1805)

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%:

```
Sub CreateViewFromActiveDocument()
'VBA801
Dim objView As HMIView
Set objView = ActiveDocument.Views.Add
objView.Zoom = 50
End Sub
```

See also

View Object (Page 1991)

PictureWindow Object (Page 1919)

4.5 AddIns

4.5.1 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

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

Loading an Addin in the Graphics Designer (Page 2417)

Example: Creating Add Ins (Page 2418)

Linking Add Ins (Page 2415)

4.5.2 Linking Add Ins

Requirements

An addin must be registered in the operating system. To register a new addin, double click on the appropriate DLL file.

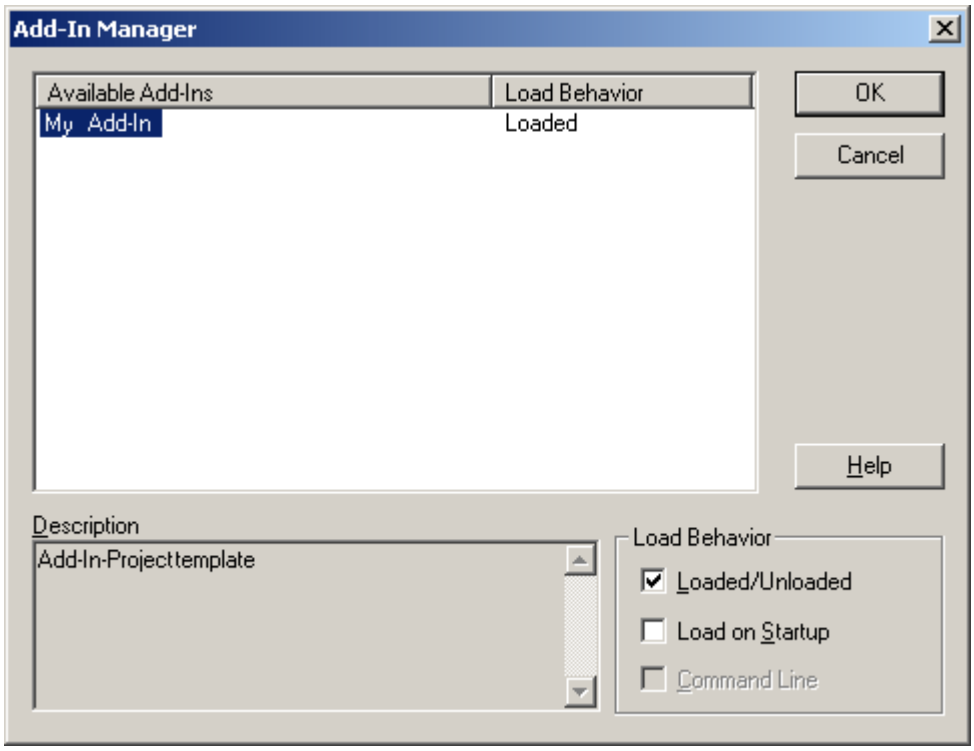
Note

For the registration to wok on a double-click, the DLL files must be linked to the application RegSvr32.exe.

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.

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 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

Loading an Addin in the Graphics Designer (Page 2417)

Example: Creating Add Ins (Page 2418)

AddIns (Page 2414)

4.5.3 Loading an Addin in the Graphics Designer

Requirements

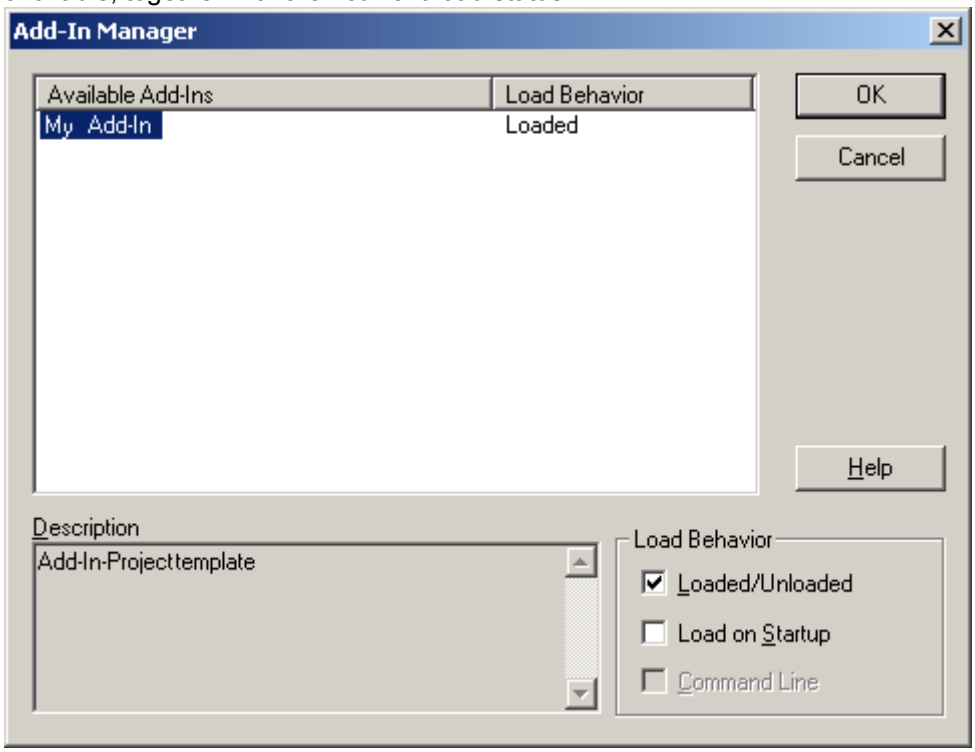
An addin must be registered in the operating system. To register a new addin, double click on the appropriate DLL file.

Note

For the registration to wok on a double-click, the DLL files must be linked to the application RegSvr32.exe.

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 2415)

AddIns (Page 2414)

4.5.4 Example: Creating Add Ins

4.5.4.1 Example: Creating Add Ins

Introduction

In order to create Addins, this documentation contains an example for Visual Basic 6.0, which creates a runable 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 2419)

Linking Add Ins (Page 2415)

4.5.4.2 Example: Creating an Add In with Visual Basic 6.0

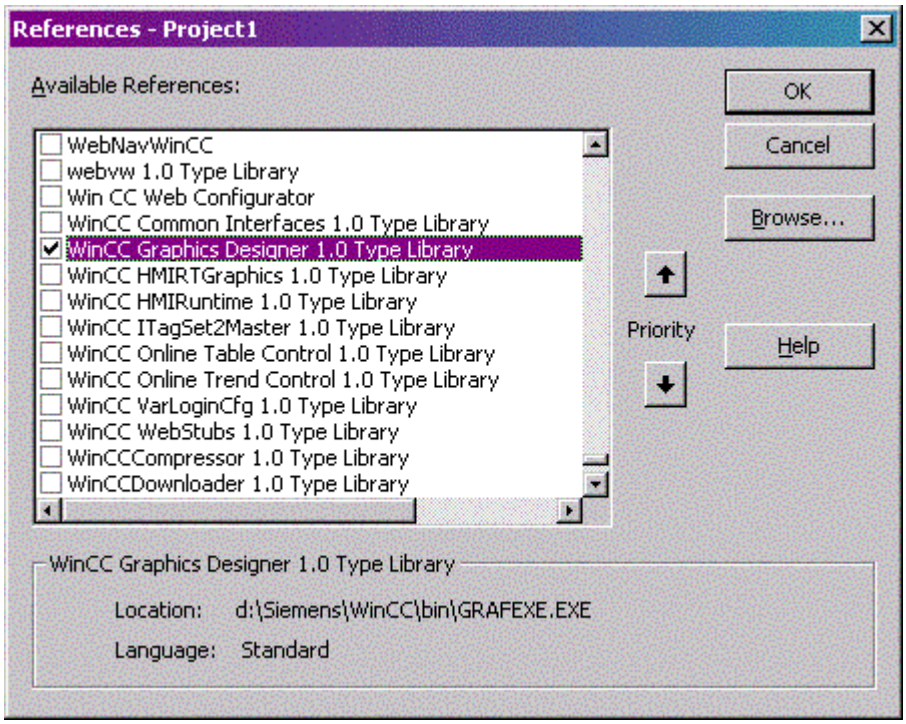
Introduction

The program code in this example produces a file called "MyAddIn.DLL". So that the addin will work in Graphics Designer, you must enable the "Load on Startup" checkbox for this addin in the Addin Manager of Graphics Designer. For this purpose you can also use the "LoadOnStartup" function in the addin.

When you open Graphics Designer the addin generates a user-defined menu. You can also use the "Tools" > "AddIns" menu to call the function contained in the addin.

Requirements

So that you can create an executable addin from the sample code, "MS Visual Studio 6.0" must You must also have referenced "WinCC Graphics Designer 1.0 Type Library" in "MS Visual Studio 6.0":



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.
3. Under "Application", select the entry for "Graphics Designer" and select the "Initial Behavior on Loading" for the addin. Close the "Connect (AddIn Designer)" dialog.

4. In Project Explorer, open the "Designer" folder and use the pop-up menu to select the command "Display Code" for the "Connect" entry.

5. Replace the entire program code with the following program code:

```
Option Explicit
'-----------------------------------------------------
'Member Variables
'-----------------------------------------------------
'Reference to the AddIn Connection
Dim WithEvents ThisAddIn As grafexe.AddinHook
'Reference to the Grafics Designer Application
Dim WithEvents GrafApp As grafexe.Application
'-----------------------------------------------------
'WithEvents AddInInstance IDTExtensibility2 (automatic)
'-----------------------------------------------------
'-----------------------------------------------------
'This method connects the AddIn 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 AddIn 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 : AddIn is not shown in the AddIn-
Menu
        '                       (ThisAddIn_Execute() is NOT called)
        ' sbAddInFilterNone    : AddIn 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
```

```
    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 AddIn functions
available in Graphics Designer
'
'By adding an application menu in Graphics Designer the menu click
events can be catched by
'the MenuItemClicked event from the application object
'------------------------------------------------------------------
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()
```

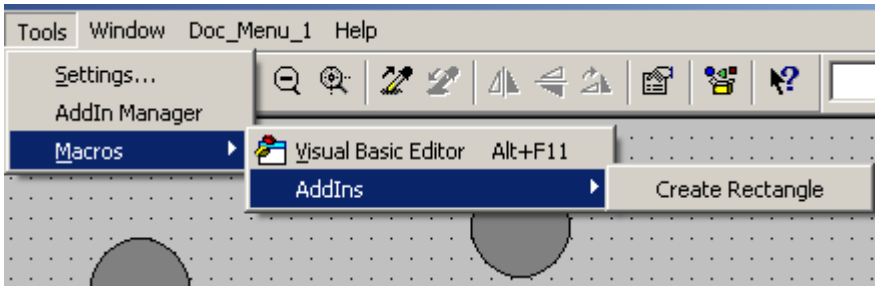
```
    Dim objDocMenu As HMIMenu
    Dim objMenuItem As HMIMenuItem

    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
item 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\Macros\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 addin, and load it in the Graphics Designer.

Result

The next time you open it, Graphics Designer will contain a user-defined menu called "DocMenu1". The menu "Tools" > "AddIns" contains an entry called "Create Rectangle", which pastes a rectangle into the active picture:



See also

Loading an Addin in the Graphics Designer (Page 2417)

Example: Creating Add Ins (Page 2418)

4.6 VBA in Other WinCC Editors

4.6.1 VBA in Other WinCC Editors

Introduction

VBA allows you to access other WinCC editors such as Tag Logging. The following editors can be automated with VBA in addition to Graphics Designer:

- Tag Management
- Tag Logging

- Text library
- Alarm Logging

The functions for accessing the editors are contained in the HMIGO class.

Requirements

The "HMIGenObjects.dll" file is referenced. This happens automatically during WinCC installation.

Principle

To enable you to have access to the HMIGO class with VBA, you must reference the "HMI GeneralObjects 1.0 Type Library" in the VBA editor ("Project" > "References"). You must create a new instance of this class in the program code, e.g.:

```
'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.

Usage

The functions provided by the HMIGO class provided you with access to Tag Management, Tag Logging, the Text Library, and Alarm Logging. To enable you to use the functions in VBA, you must have opened a project in WinCC. In addition, you can also access the properties of the class directly.

In this way, directly from the program code you can, for example, created several tags and change their values, edit text entries in the TextLibrary, or customize 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 e.g. the instruction HMIGO = nothing.

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:

```
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
```

As a result, an error text returned by the interface is output.

See also

VBA in Alarm Logging (Page 2478)

VBA in the Text Library (Page 2465)

VBA in Tag Logging (Page 2438)

VBA in Tag Management (Page 2426)

4.6.2 VBA in Tag Management

4.6.2.1 VBA in Tag Management

Introduction

VBA allows you to create tags directly from the program code, modify and delete them, and read out and change their properties, types, and values.

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 enumerations "HMIGO_TAG_TYPE" and "HMIGO_TAG_LIST_TYPE" are available for the parameter assignment of these functions.

Note

You should not have or should not open the tag 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.

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:

Object property	Description	Read/Write
ObjectStateTag	Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under "VBA in other WinCC Editors".	Yes/no
TagName	Name of the tag	Yes/no
TagGroupName	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.	Yes/no
TagConnection	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.	Yes/no
TagMaximum	Sets the new value of the upper limit	Yes/yes
TagMinimum	Sets the new value of the lower limit	Yes/yes
TagStart	Sets the new start value	Yes/yes
TagS5S7Address	Address of the S7 or S5 PLC to which the tag is connected. If no address is specified, a blank entry is passed.	Yes/yes

Object property	Description	Read/Write
TagType (Enum)	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 (Enum)	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 test)
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
UseSubstValueOnCommonError	Set the replacement value for connection errors.	Yes/yes
UseSubstValueOnMaxLimit	Set the replacement value for upper limit.	Yes/yes
UseSubstValueOnMinLimit	Set the replacement value for lower limit.	Yes/yes
UseSubstValueOnStartValue	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 2436)

GetTag Function (Page 2435)

DeleteTag Function (Page 2434)

CreateTag Function (Page 2432)

CommitTag Function (Page 2430)

CloseTag Function (Page 2429)

VBA in Other WinCC Editors (Page 2424)

4.6.2.2 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 2436)

GetTag Function (Page 2435)

DeleteTag Function (Page 2434)

CreateTag Function (Page 2432)

CommitTag Function (Page 2430)

VBA in Tag Management (Page 2426)

4.6.2.3 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
' 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"
 '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 2436)

GetTag Function (Page 2435)

DeleteTag Function (Page 2434)

CreateTag Function (Page 2432)

CloseTag Function (Page 2429)

VBA in Tag Management (Page 2426)

4.6.2.4 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

Parameter (Data Type)	Description
TagName (string)	Name of the tag to be created.
TagType (HMIGO_TAG_TYPE)	Data type of the tag. The possible types are: TAG_BINARY_TAG TAG_SIGNED_8BIT_VALUE TAG_UNSIGNED_8BIT_VALUE TAG_SIGNED_16BIT_VALUE TAG_UNSIGNED_16BIT_VALUE TAG_SIGNED_32BIT_VALUE TAG_UNSIGNED_32BIT_VALUE TAG_FLOATINGPOINT_NUMBER_32BIT_IEEE_754 TAG_FLOATINGPOINT_NUMBER_64BIT_IEEE_754 TAG_TEXT_TAG_8BIT_CHARACTER_SET TAG_TEXT_TAG_16BIT_CHARACTER_SET TAG_RAW_DATA_TYPE TAG_TEXT_REFERENCE

Parameter (Data Type)	Description
Connection (String, optional)	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.
S7S5Address (String, optional)	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.
GroupName (String, optional)	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.

Example:

```
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 2436)

GetTag Function (Page 2435)

DeleteTag Function (Page 2434)

CommitTag Function (Page 2430)

CloseTag Function (Page 2429)

VBA in Tag Management (Page 2426)

4.6.2.5 DeleteTag Function

Description

Deletes the specified tag.

syntax

```
Expression.DeleteTag(TagName)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
TagName (string)	Name of the tag to be deleted.

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 2436)

GetTag Function (Page 2435)

CreateTag Function (Page 2432)

CommitTag Function (Page 2430)

CloseTag Function (Page 2429)

VBA in Tag Management (Page 2426)

4.6.2.6 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

Parameter (Data Type)	Description
TagName (string)	Name of the tag whose values are to be read in.

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 2436)

DeleteTag Function (Page 2434)

CreateTag Function (Page 2432)

CommitTag Function (Page 2430)

CloseTag Function (Page 2429)

VBA in Tag Management (Page 2426)

4.6.2.7 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

Parameter (Data Type)	Description
ListType (HMIGO_TAG_LIST_TYPE)	Defines which content should be returned as a list. Possibilities are: • TAG_CHANNEL_UNITS (0) all channel units created • TAG_CHANNELS (2) all channels created • TAG_CONNECTIONS (3) all connections created • TAG_GROUPS (4) all tag groups created • TAG_NAMES (5) all tags created
pListArray (Variant)	List with the requested content.
Filter (String)	Filters can be set optionally. Wildcards "*" and "?" are also possible.

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:

```
Sub ReadTagByConnection()
'HMIGO_027
'read content in data manager by connections
'no conections 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 2435)

DeleteTag Function (Page 2434)

CreateTag Function (Page 2432)

CommitTag Function (Page 2430)

CloseTag Function (Page 2429)

VBA in Tag Management (Page 2426)

4.6.3 VBA im Tag Logging

4.6.3.1 VBA in Tag Logging

Introduction

VBA allows you to create process value archives and archive tags directly from the program code, modify them, and delete them.

Principle

When you have created the instance of the HMIGO class, the following functions are available to you to access Tag Logging:

- CloseTlgArchive
- CloseTlgTag
- CommitTlgArchive
- CommitTlgTag
- CreateTlgArchive
- CreateTlgTag
- DeleteTlgArchive
- DeleteTlgTag
- GetTlgArchive
- GetTlgTag
- ListTlgArchive
- ListTlgTag

Note

Tag Logging must not be open or be opened when editing with VBA.
Before you use the trigger of the tag logging in VBA with a newly created WinCC project, you must first start the Editor "Tag Logging", create an archive and then again close the Editor. The triggers are created by the Editor.

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).

Object property	Description	Read/Write	is used in
ObjectStateTlgArchive	Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under "VBA in other WinCC Editors".	Yes/no	P, V
TlgArchiveName	Name of the process value archive or compressed archive.	Yes/no	P, V
TlgArchiveType	Specifies whether the archive is a process value archive or a compressed archive.	Yes/no	P, V
TlgArchiveAccessLevel Read	The authorization level for reading.	Yes/no	P, V
TlgArchiveAccessLevel Write	The authorization level for writing.	Yes/no	P, V
TlgArchiveArchiveState	Specifies whether archiving is disabled or enabled at system startup. Possible values of 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 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 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

Object property	Description	Read/Write	is used in
ObjectStateTlgTag	Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under "VBA in other WinCC Editors".	Yes/no	P, V
TlgTagArchiveName	Name of the archive tag.	Yes/no	P, V
TlgTagName	Name of the archive.	Yes/no	P, V
TlgTagType	Specifies the tag type. The possible types are in Enum HMIGO_TLG_TAG_TYPE. • TLG_TAG_TYP_ANALOG (65537) • TLG_TAG_TYP_BINARY (65538) • TLG_TAG_TYP_PROCESS (65544) • TLG_TAG_TYP_COMPRESS (65540)	Yes/yes	P, V
TlgTagArchiving	Specifies the acquisition type. Possible values of Enum HMIGO_TLG_TAG_ARCHIVING: • TLG_TAG_ACYCLIC (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 Enum HMIGO_TLG_TAG_ARCHIVING_STATE: • TLG_TAG_LOCKED (1) • TLG_TAG_ACTIVATED (0)	Yes/yes	P, V
TlgTagOnError	Specifies whether, in the event of a problem, the most recently acquired value or the substitute value is saved. Possible values of Enum HMIGO_TLG_TAG_ON_ERROR: • TLG_TAG_LAST_VALUE (1) • TLG_TAG_SUBSTITUTE (2)	Yes/yes	P
TlgTagTriggerType	Specifies how archiving is carried out at a signal change. Possible values of Enum HMIGO_TLG_TAG_TRIGGER_TYPE: • TLG_TAG_FROM_0_TO_1 (2) • TLG_TAG_FROM_1_TO_0 (3) • TLG_TAG_ALWAYS (4) • TLG_TAG_EVERY_CHANGE (1)	Yes/yes	P

Object property	Description	Read/Write	is used in
TlgTagOnChange	Specifies whether archiving is to be carried out in the event of a change. Possible values of Enum HMIGO_TLG_TAG_ON_CHANGE: • TLG_TAG_EVERY_VALUE (0) • TLG_TAG_RELATIVE_HYSTERESE (1) • TLG_TAG_ABSOLUTE_HYSTERESE (2)	Yes/yes	P
TlgTagMethodType	Specifies the method by which the value is edited before archiving. Possible values of Enum HMIGO_TLG_TAG_METHOD_TYPE: • TLG_TAG_ACTUAL (1) • TLG_TAG_SUM (3) • TLG_TAG_MaxValue (5) • TLG_TAG_MinValue (4) • TLG_TAG_AVERAGE (2)	Yes/yes	P, V
TlgTagTriggerScan	Name of the timer for the acquisition cycle	Yes/yes	P
TlgTagTriggerArchiving	Name of the timer for the archiving cycle.	Yes/yes	P
TlgTagStartTriggerFunction	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.	Yes/yes	P
TlgTagStartTriggerModule	Specifies the name of a DLL from which the script function is called for the checking of a start event.	Yes/yes	P
TlgTagStopTriggerFunction	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.	Yes/yes	P
TlgTagTriggerFunction	Specifies the name of a script function for the dynamic switching of the acquisition and archiving cycles.	Yes/yes	P
TlgTagNameCompressArchive	In the case of compressed archives, contains the name of the source archive.	Yes/yes	V
TlgTagNameCompressTag	In the case of compressed archives, contains the name of the source tag.	Yes/yes	V
TlgTagNameRawValue	In the cast of process-controlled archives, contains the name of the raw-data tag.	Yes/yes	P
TlgTagConvertModule	Name of the conversion DLL used for data conversion.	Yes/yes	P
TlgTagNameProcTag	Name of the process tag from which the value to be acquired is taken	Yes/yes	P
TlgTagAliasName	The alternative name by means of which the tag can be addressed (alias).	Yes/yes	P
TlgTagStartEvent	Name of the tag by means of which the start of archiving is checked.	Yes/yes	P
TlgTagStopEvent	Name of the tag by means of which the stopping of archiving is checked.	Yes/yes	P
TlgTagTriggerFactor	Contains the factor for the display cycle as a multiple of the archiving cycle.	Yes/yes	P
TlgTagUpperLimit	Value for the scaling of the tag's upper limit.	Yes/yes	P

Object property	Description	Read/Write	is used in
TlgTagLowerLimit	Value for the scaling of the tag's lower limit.	Yes/yes	P
TlgTagHysterese	Value for the hysteresis by means of which a check is carried out to establish whether a value has changed.	Yes/yes	P

See also

ListTlgTag Function (Page 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Other WinCC Editors (Page 2424)

4.6.3.2 CloseTlgArchive-Funktion

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 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

VBA in Tag Logging (Page 2438)

4.6.3.3 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
```

See also

ListTlgTag Function (Page 2463)
ListTlgArchive Function (Page 2461)
GetTlgArchive Function (Page 2458)
DeleteTlgTag Function (Page 2457)
DeleteTlgArchive Function (Page 2455)
CreateTlgTag Function (Page 2451)
CreateTlgArchive Function (Page 2448)
CommitTlgTag Function (Page 2446)
CommitTlgArchive Function (Page 2445)
CloseTlgArchive-Funktion (Page 2442)
VBA in Tag Logging (Page 2438)

4.6.3.4 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 2455)

ListTlgTag Function (Page 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.3.5 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 2463)
ListTlgArchive Function (Page 2461)
GetTlgArchive Function (Page 2458)
DeleteTlgTag Function (Page 2457)
DeleteTlgArchive Function (Page 2455)
CreateTlgTag Function (Page 2451)
CreateTlgArchive Function (Page 2448)
CommitTlgTag Function (Page 2446)
CommitTlgArchive Function (Page 2445)
CloseTlgTag Function (Page 2443)
CloseTlgArchive-Funktion (Page 2442)
VBA in Tag Logging (Page 2438)

4.6.3.6 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

Parameter (Data Type)	Description
ArchiveName (String)	Name of the archive to be created
ArchiveType (HMIGO_TLG_ARCHIVE_TYPE)	Type of the archive. The possible types are: • TLG_PROCESSARCHIVE (131073) for a process value archive • TLG_COMPRESSARCHIVE (131074) for a compressed archive

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.

Property	Default Value (Enum Name => Value)	Comment
TlgArchiveAccessLevelRead	0	Without authorization level
TlgArchiveAccessLevelWrite	0	Without authorization level
TlgArchiveArchiveState	TLG_ARCHIVE_STATE_ACTIVATED (0)	Archiving is started at start of Runtime.
TlgArchiveBufferSize	1000	Number of data records
TlgArchiveBufferType	TLG_ARCHIVE_BUFFER_TYPE_DISK (2)	The values are stored on hard disk in the database.
TlgArchiveCompressRange	"1 Tag". This string must be created individually for each language (e.g. English: "1 day")	Corresponds to exactly one day. Only relevant in the case of compressed tags. Special Feature: the user is responsible for values >= 1 minute
TlgArchiveCompressType	TLG_COMPRESS_TYPE_CALC (1)	Only calculate compression values. Only relevant in the case of compressed tags.

Enum HMIGO_TLG_ARCHIVE_STATE

Parameters	Description
TLG_ARCHIVE_STATE_LOCKED (1)	Archiving is disabled at system startup.
TLG_ARCHIVE_STATE_ACTIVATED (0)	Archiving is started at start of Runtime.

Enum HMIGO_TLG_ARCHIVE_BUFFER_TYPE

Parameters	Description
TLG_ARCHIVE_BUFFER_TYPE_DISK (2)	The values are archived on hard disk.
TLG_ARCHIVE_BUFFER_TYPE_RAM (1)	The values are only archived in working memory.

Enum HMIGO_TLG_ARCHIVE_COMPRESS_TYPE

Parameters	Description
TLG_COMPRESS_TYPE_CALC (1)	Only the compression values are calculated.
TLG_COMPRESS_TYPE_CALC_COPY (2)	The compression values are calculated and the original values copied.
TLG_COMPRESS_TYPE_CALC_DEL (3)	The compression values are calculated and the original values then deleted.
TLG_COMPRESS_TYPE_CALC_COPY_DEL (4)	The compression values are calculated and the original values copied and then deleted.

Example:

```
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 2458)

ListTlgTag Function (Page 2463)

ListTlgArchive Function (Page 2461)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgTag Function (Page 2451)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.3.7 CreateTlgTag Function

Description

Creates a new archive tag.

syntax

```
Expression.CreateTlgTag(ArchiveName,TagName,[TagType])
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
ArchiveName (String)	Name of an existing archive in which the tag is to be entered
TagName (string)	Name of the tag to be created
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. The possible types are: • TLG_VAR_TYP_ANALOG (65537) for an analog tag. • TLG_VAT_TYP_BINARY (65538) for a binary tag. • TLG_VAR_TYP_PROCESS (65544) for a process tag. • TLG_VAT_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.

Property	Default Value (Enum Name => Value)	Comment
TlgTagType	TLG_VAR_TYP_ANALOG (65537)	Acquired by means of an analog data manager tag
TlgTagArchiving	TLG_TAG_CYCLIC_CONTINUOUS (8388610)	Cyclic, continuous acquisition
TlgTagArchivingState	TLG_TAG_ACTIVATED (0)	Archiving is started at start of Runtime.

Property	Default Value (Enum Name => Value)	Comment
TlgTagTriggerScan	1 second	Please note that "1 second" is only the name of the trigger. You must ensure yourself that the trigger exists and actually has a cycle of 1 s.
TlgTagTriggerArchiving	1 second	Please note that "1 second" is only the name of the trigger. You must ensure yourself that the trigger exists and actually has a cycle of 1 s.
TlgTagTriggerFactor	1	The display cycle and archiving cycle are identical.
TlgTagOnError	TLG_TAG_LAST_VALUE (1)	The last valid value is taken as the substitute value.
TlgTagTriggerType	TLG_TAG_ALWAYS (4)	Every value is archived.
TlgTagMethodType	TLG_TAG_ACTUAL (1)	No editing. The value is accepted immediately.
TlgTagStartTriggerFunction	No function specified	--
TlgTagStopTriggerFunction	No function specified	--
TlgTagTriggerFunction	No function specified	--
TlgTagUpperLimit	No value specified	--
TlgTagLowerLimit	No value specified	--
TlgTagNameCompressArchive	No archive name specified	--
TlgTagNameCompressTag	No tag name specified	--
TlgTagNameRawValue	No raw-data tag specified	--
TlgTagStartTriggerModule	No DLL name specified	--
TlgTagNameProcTag	Corresponds to "TagName"	--
TlgTagOnChange	TLG_TAG_EVERY_VALUE (0)	Every value will be archived.
TlgTagHysterese	0	No check is carried out by means of hysteresis.
TlgTagAliasName	No value specified	--
TlgTagStartEvent	No tag specified	--
TlgTagStopEvent	No tag specified	--

List of the enumerators for Tag Logging

Enum types	Description
TLG_TAG_TYPE	The passed parameter specifies the type of the tag. The possible types are in the table Enum HMIGO_TLG_TAG_TYPE.
TLG_TAG_ARCHIVING	The passed parameter specifies the acquisition type. The possible values are in the table Enum HMIGO_TLG_TAG_ARCHIVING.

Enum types	Description
TLG_TAG_ARCHIVING_STATE	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.
TLG_TAG_ON_ERROR	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.
TLG_TAG_TRIGGER_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_TRIGGER_TYPE.
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

Values	Description
TLG_TAG_TYP_ANALOG (65537)	Analog tag
TLG_TAG_TYP_BINARY (65538)	Binary Tags
TLG_TAG_TYP_PROCESS (65544)	Process tag
TLG_TAG_TYP_COMPRESS (65540)	Compressed archive tag

Enum HMIGO_TLG_TAG_ARCHIVING

Values	Description
TLG_TAG_ACYCLIC (8388609)	Acyclic acquisition
TLG_TAG_CYCLIC_CONTINUOUS (8388610)	Cyclic-continuous acquisition
TLG_TAG_CYCLIC_SELECTIVE (8388612)	Cyclic-selective acquisition
TLG_TAG_ON_EVERY_CHANGE (8388616)	Acquisition only in the event of a change

Enum HMIGO_TLG_TAG_ARCHIVING_STATE

Values	Description
TLG_TAG_LOCKED (1)	Acquisition disabled at system startup
TLG_TAG_ACTIVATED (0)	Acquisition enabled at system startup

Enum HMIGO_TLG_TAG_ON_ERROR

Values	Description
TLG_TAG_LAST_VALUE (1)	The most recently acquired value is used.
TLG_TAG_SUBSTITUTE (2)	A substitute value is entered.

Enum HMIGO_TLG_TAG_TRIGGER_TYPE

Values	Description
TLG_TAG_FROM_0_TO_1 (2)	Signal change from the value 0 to 1
TLG_TAG_FROM_1_TO_0 (3)	Signal change from the value 1 to 0
TLG_TAG_ALWAYS (4)	Always archive.
TLG_TAG_EVERY_CHANGE (1)	Archive at every signal change.

Enum HMIGO_TLG_TAG_METHOD_TYPE

Values	Description
TLG_TAG_ACTUAL (1)	The current value is accepted.
TLG_TAG_SUM (3)	The sum is formed.
TLG_TAG_MaxValue (5)	The greatest value is saved.
TLG_TAG_MinValue (4)	The smallest value is saved.
TLG_TAG_AVERAGE (2)	The average value is saved.

Enum HMIGO_TLG_TAG_ON_CHANGE

Values	Description
TLG_TAG_EVERY_VALUE (0)	The current value is accepted.
TLG_TAG_RELATIVE_HYSTERESE (1)	A hysteresis specified as a percentage is used for the calculation as to whether the value is to be archived.
TLG_TAG_ABSOLUTE_HYSTERESE (2)	A hysteresis specified as an absolute value is used for the calculation as to whether the value is to be archived.

Example:

```
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 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CloseTlgArchive-Funktion (Page 2442)

CloseTlgTag Function (Page 2443)

VBA in Tag Logging (Page 2438)

4.6.3.8 DeleteTlgArchive Function

Description

Deletes the specified archive.

Syntax

```
Expression.DeleteTlgArchive(ArchiveName)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
ArchiveName (String)	Name of the archive to be deleted. Archive tags contained in the archive are also deleted.

Example:

```
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 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.3.9 DeleteTlgTag Function

Description

Deletes the specified archive tag.

syntax

```
Expression.DeleteTlgTag(ArchiveName,TagName)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
ArchiveName (String)	Name of the archive containing the archive tag to be deleted
TagName (string)	Name of the archive tag to be deleted.

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 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgArchive Function (Page 2455)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.3.10 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

Parameter (Data Type)	Description
ArchiveName (String)	Name of the archive whose values are to be read in.

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 2451)

ListTlgTag Function (Page 2463)

ListTlgArchive Function (Page 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.3.11 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

Parameter (Data Type)	Description
ArchiveName (String)	Name of the archive containing the archive tag.
TagName	Name of the archive tag whose parameters are to be read in.

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 2451)
ListTlgTag Function (Page 2463)
ListTlgArchive Function (Page 2461)
GetTlgArchive Function (Page 2458)
DeleteTlgTag Function (Page 2457)
DeleteTlgArchive Function (Page 2455)
CreateTlgArchive Function (Page 2448)
CommitTlgTag Function (Page 2446)
CommitTlgArchive Function (Page 2445)
CloseTlgTag Function (Page 2443)
CloseTlgArchive-Funktion (Page 2442)
VBA in Tag Logging (Page 2438)

4.6.3.12 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

Parameter (Data Type)	Description
ListType (HMIGO_TLG_ARCHIVE_LIST_TYPE)	Defines which content should be returned in a list. Possibilities are: • TLG_ARCHIVE_NAMES (1) All created Tag Logging archives • TLG_ARCHIVE_TRIGGER_NAMES (2) All created cycles / timers
pListArray (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.

Example:

In the following example, a check is made whether archives are configured:

```
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 2463)
GetTlgArchive Function (Page 2458)
DeleteTlgTag Function (Page 2457)
DeleteTlgArchive Function (Page 2455)
CreateTlgTag Function (Page 2451)
CreateTlgArchive Function (Page 2448)
CommitTlgTag Function (Page 2446)
CommitTlgArchive Function (Page 2445)
CloseTlgTag Function (Page 2443)
CloseTlgArchive-Funktion (Page 2442)
VBA in Tag Logging (Page 2438)

4.6.3.13 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

Parameter (Data Type)	Description
ListType (HMIGO_TLG_TAG_LIST_TYPE)	Defines which content should be returned in a list. Possibilities are: TLG_TG_NAMES (1) All tags created in a Tag Logging archive
ListArray (Variant)	List with the requested content.
ArchiveName (String)	Name of the archive in Tag Logging (optional). If the name of the archive is not specified, all,archive tags are returned.
Filter (String)	Filters can be set optionally. Wildcards "*" and "?" are also possible.

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 2461)

GetTlgArchive Function (Page 2458)

DeleteTlgTag Function (Page 2457)

DeleteTlgArchive Function (Page 2455)

CreateTlgTag Function (Page 2451)

CreateTlgArchive Function (Page 2448)

CommitTlgTag Function (Page 2446)

CommitTlgArchive Function (Page 2445)

CloseTlgTag Function (Page 2443)

CloseTlgArchive-Funktion (Page 2442)

VBA in Tag Logging (Page 2438)

4.6.4 VBA in the Text Library

4.6.4.1 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.

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
- ListText
- ModifyText

The enumerations "HMIGO_TEXT_CREATE_MODE" and "HMIGO_TEXT_LIST_TYPE"are available for the parameter assignment of these functions.

Note

The TextLibrary must not be open or be opened when editing using VBA.

See also

ModifyText Function (Page 2476)
ListText Function (Page 2475)
GetTextID Function (Page 2473)
GetText Function (Page 2471)
DeleteTextLanguage Function (Page 2470)
DeleteText Function (Page 2468)
CreateText Function (Page 2467)
CreateTextLanguage Function (Page 2466)
VBA in Other WinCC Editors (Page 2424)

4.6.4.2 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

Parameter (Data Type)	Description
LanguageID (Long)	ID for the language to be created (e.g. 1031 for German, 1033 for English, etc.) For a table of all language codes refer to the WinCC online help on "Language Identifiers".

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 lngLangugeNumber As Long
  Set objHMIGO = New HMIGO
  lngLangugeNumber = 1065     'farsi
 'create new language
  objHMIGO.CreateTextLanguage lngLangugeNumber
  Set objHMIGO = Nothing
End Sub
```

See also

ModifyText Function (Page 2476)

ListText Function (Page 2475)

GetTextID Function (Page 2473)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateText Function (Page 2467)

VBA in the Text Library (Page 2465)

4.6.4.3 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

Parameter (Data Type)	Description
LanguageID (Long)	ID of the language for which the text will be created.
Text (string)	Text to be created.
CreateMode (HMIGO_TEXT_CREATE_MODE)	Mode of text creation: • TEXT_ADD_REFCOUNT (0) only increases the reference counter when an identical text already exists. • TEXT_CREATE_ALWAYS (1) always sets up a new text line and inserts the text in it.
TextID (long)	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.

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
  Set objHMIGO = New HMIGO
    strText = "new text"
  '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 2476)

ListText Function (Page 2475)

GetTextID Function (Page 2473)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.4.4 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

Parameter (Data Type)	Description
TextID (long)	ID of the line of text to be deleted

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 ocurred"
  'reset errorhandler
  Err.Clear
End Sub
```

See also

VBA in the Text Library (Page 2465)

ModifyText Function (Page 2476)

ListText Function (Page 2475)

GetTextID Function (Page 2473)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

CreateText Function (Page 2467)

CreateTextLanguage Function (Page 2466)

4.6.4.5 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

Parameter (Data Type)	Description
LanguageID (Long)	ID of the language to be deleted

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 2471)

ModifyText Function (Page 2476)

ListText Function (Page 2475)

GetTextID Function (Page 2473)

DeleteText Function (Page 2468)

CreateText Function (Page 2467)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.4.6 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

Parameter (Data Type)	Description
LanguageID (Long)	ID of the language of the text to be read
TextID (long)	ID of the line of text from which text is to be read
Text (string)	Returns the text of the selected line of text and language.

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 2476)

ListText Function (Page 2475)

GetTextID Function (Page 2473)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateText Function (Page 2467)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.4.7 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

Parameter (Data Type)	Description
LanguageID (Long)	ID of the language of the text searched for
Text (string)	The text searched for
TextID (long)	ID of the line of text in which the text searched for was found

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 & """ found in TextID: " & 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 ocurred"
  'reset errorhandler
Err.Clear
End Sub
```

See also

ModifyText Function (Page 2476)

ListText Function (Page 2475)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateText Function (Page 2467)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.4.8 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

```
Exoression.ListText(ListType,pListArray,[LanguageID],[Filter])
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
ListType (HMIGO_TEXT_LIST_TYPE)	Defines which content should be returned as a list. Possibilities are: • TEXT_LANGUAGE_IDS (1) All the created languages. The result still has to be converted by adding 400hex. • TEXT_IDS (2) All text IDs. • TEXT_TEXTS (3) All texts in a language.
pListArray (Variant)	List with the requested content.
LanguageID (Long)	The language ID whose text is to be returned.
Filter (String)	Filters can be set optionally. Wildcards "*" and "?" are also possible.

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 2476)

GetTextID Function (Page 2473)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.4.9 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

Parameter (Data Type)	Description
LanguageID (Long)	ID of the language of the text to be changed.
TextID (long)	ID of the language of the text to be changed.
Text (string)	New text to be inserted.

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 & """ found in TextID: " & lngTextID & vbNewLine & _
           "Text replaced with : """ & strNewText & """ !", vbOKOnly, "Result DeleteText"
Else
    MsgBox "Text : """ & strOldText & """ not found." & vbNewLine & _
           "No Replacements done!", 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 ocurred"
  'reset errorhandler
Err.Clear
End Sub
```

See also

ListText Function (Page 2475)

GetTextID Function (Page 2473)

GetText Function (Page 2471)

DeleteTextLanguage Function (Page 2470)

DeleteText Function (Page 2468)

CreateText Function (Page 2467)

CreateTextLanguage Function (Page 2466)

VBA in the Text Library (Page 2465)

4.6.5 VBA in Alarm Logging

4.6.5.1 VBA in Alarm Logging

Introduction

VBA allows you to create messages directly from the program code, modify them, and delete them.

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

Enumeration "HMIGO_SINGLE_ALARM_CLASS_IDS" and "HMIGO_SINGLE_ALARM_LIST_TYPE" is available for parameter assignment of this function.

Note

You should not have or should not open the Alarm Logging when editing with VBA.

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:

Object property	Description	Read/Write
ObjectStateSingleAlarm	Returns the object state via the enumeration HMIGO_OBJECT_STATE. Further information on this enumeration can be found in this documentation under "VBA in other WinCC Editors".	Yes/no
SingleAlarmMessageNumber	Number of the message	Yes/no
SingleAlarmAGNumber	AS Number	Yes/yes
SingleAlarmCPUNumber	CPU number of the AGs.	Yes/yes
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: Values from 263 to 288	Yes/yes
SingleAlarmTextXXID XX = 1...10	The properties SingleAlarmText1ID to SingleAlarmText10ID exist for the user texts 1 to 10.	Yes/yes

Object property	Description	Read/Write
SingleAlarmTagNameProcessValueXX XX = 1...10	For the process values there are the properties SingleAlarmTagNameProcessValue1 through 10 If you want to delete a configured process value, you must describe this parameter with a tag of the type "Long", which has the value "0".[1)]	Yes/yes
SingleAlarmTagName	Tag name for event	Yes/yes
SingleAlarmMessageBit	Bits for bit reporting procedure	Yes/yes
SingleAlarmQuitTag	Tag name for acknowledgment status	Yes/yes
SingleAlarmQuitBits	Bit for bit reporting procedure	Yes/yes
SingleAlarmStateTag	Tag for status query	Yes/yes
SingleAlarmStateBits	Bit for status tag	Yes/yes
SingleAlarmNormDLL	Name of the conversion DLL	Yes/yes
SingleAlarmQuitSingle	Acknowledgment of the messages, TRUE or FALSE possible	Yes/yes
SingleAlarmHornActivate	Activation of the horn, TRUE or FALSE possible	Yes/yes
SingleAlarmArchiving	Archiving of the message, TRUE or FALSE possible	Yes/yes
SingleAlarmProtocol	Logging of the message, TRUE or FALSE possible	Yes/yes
SingleAlarmFlankInvert	Triggering of message at falling edge, TRUE or FALSE possible	Yes/yes
SingleAlarmLockedOnStart	Message is disabled at system startup, TRUE or FALSE possible	Yes/yes
SingleAlarmGlobalAPFunction	Forward message to global AP function, TRUE or FALSE possible	Yes/yes
SingleAlarmActionName	Name of the action	Yes/yes
SingleAlarmActionParams	Parameters of the action	Yes/yes
SingleAlarmInfoText	Information text for message	Yes/yes
SingleAlarmGroup	Name of the user-defined group message assigned to a message.	Yes/yes

1)

```
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 2489)

GetSingleAlarm Function (Page 2488)

DeleteSingleAlarm Function (Page 2487)

CreateSingleAlarm Function (Page 2483)

CommitSingleAlarm Function (Page 2482)

CloseSingleAlarm Function (Page 2481)

VBA in Other WinCC Editors (Page 2424)

4.6.5.2 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 2489)

GetSingleAlarm Function (Page 2488)

DeleteSingleAlarm Function (Page 2487)

CreateSingleAlarm Function (Page 2483)

CommitSingleAlarm Function (Page 2482)

VBA in Alarm Logging (Page 2478)

4.6.5.3 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 2489)

GetSingleAlarm Function (Page 2488)

DeleteSingleAlarm Function (Page 2487)

CreateSingleAlarm Function (Page 2483)

CloseSingleAlarm Function (Page 2481)

VBA in Alarm Logging (Page 2478)

4.6.5.4 CreateSingleAlarm Function

Description

Creates a new message.

syntax

```
Expression.CreateSingleAlarm(MessageNumber,ClassID,MessageTypeID,Tex
t1ID,MessageTagName,MessageBit)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
MessageNumber (long)	Number of the message. • If an unused message number is specified here, it is accepted. • If the value "0" is entered, the message number is freely assigned by the system. In this case, the message number is given back here.
ClassID (HMIGO_SINGLE_ALARM_CLASS_IDS)	Message class. The possible values are in the table Enum HMIGO_SINGLE_ALARM_CLASS_IDS.
MessageTypeID (Integer)	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: Values from 263 to 288
Text1ID (Long)	ID for the first user text. The ModifySingleAlarm function can be used to define nine further user texts (1-10).
MessageTagName (String)	Tag name for the event.
MessageBit (integer)	Bit in bit reporting process (0...31)

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.

Parameters	Default Value (Enum Name => Value)	Comment
SingleAlarmAGNumber	0	--
SingleAlarmCPUNumber	0	--
SingleAlarmTextXXID	No text entered	--
SingleAlarmTagNameProcessValueXX	No tag entered	--

Parameters	Default Value (Enum Name => Value)	Comment
SingleAlarmQuitTag	No tag entered	--
SingleAlarmQuitBits	0	No bits set.
SingleAlarmStateTag	No tag entered	Corresponds to exactly one day. Only relevant in the case of compressed tags.
SingleAlarmStateBits	0	No bits set.
SingleAlarmNormDLL	No name entered	--
SingleAlarmQuitSingle	FALSE	Single acknowledgment, no group acknowledgment
SingleAlarmHornActivate	FALSE	Horn Not active.
SingleAlarmArchiving	TRUE	Message will be archived.
SingleAlarmProtocol	TRUE	Message is logged.
SingleAlarmFlankInvert	FALSE	Not activated.
SingleAlarmLockedOnStart	FALSE	Message is not disabled.
SingleAlarmGlobalAPIFunction	FALSE	Message is not forwarded.
SingleAlarmActionName	No name entered	--
SingleAlarmActionParams	No parameters entered for the action	--
SingleAlarmInfoText	No text entered	--
SingleAlarmGroup	No text entered	--

Enum HMIGO_SINGLE_ALARM_CLASS_IDS

The following message classes are available for selection:

Values	Description
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)	--

Values	Description
SINGLE_ALARM_CLASS_16 (16)	--
SINGLE_ALARM_CLASS_SYSTEM_REQUIRE_ACKNOWLEDGEMENT (17)	--
SINGLE_ALARM_CLASS_SYSTEM_WITHOUT_ACKNOWLEDGEMENT (18)	--

Example:

```
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 reqiured parameter
  lngMsgNumber = 50
  lngMsgClassID = 1
  lngMsgTypeID = 2
  lngMsgTextID = 69
  strMsgText = "new text message"
  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 2489)

GetSingleAlarm Function (Page 2488)

DeleteSingleAlarm Function (Page 2487)

CommitSingleAlarm Function (Page 2482)

CloseSingleAlarm Function (Page 2481)

VBA in Alarm Logging (Page 2478)

4.6.5.5 DeleteSingleAlarm Function

Description

Deletes the specified message.

syntax

```
Expression.DeleteSingleAlarm(MessageNumber)
```

Expression

Necessary. An expression which returns a "HMIGO" type object.

Parameters

Parameter (Data Type)	Description
MessageNumber (long)	Number of the message to be deleted.

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
```

See also

VBA in Alarm Logging (Page 2478)

ListSingleAlarm Function (Page 2489)

GetSingleAlarm Function (Page 2488)

CreateSingleAlarm Function (Page 2483)

CommitSingleAlarm Function (Page 2482)

CloseSingleAlarm Function (Page 2481)

4.6.5.6 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

Parameter (Data Type)	Description
MessageNumber (long)	The message number of the message to be read in.

Example:

```
Sub GetSingleAlarm()
' HMIGO_026
' 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/ 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 2489)

DeleteSingleAlarm Function (Page 2487)

CreateSingleAlarm Function (Page 2483)

CommitSingleAlarm Function (Page 2482)

CloseSingleAlarm Function (Page 2481)

VBA in Alarm Logging (Page 2478)

4.6.5.7 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

Parameter (Data Type)	Description
ListType (HMIGO_SINGLE_ALARM_LIST_TYPE)	Defines which content should be returned in a list. Possibilities are: • SINGLE_ALARM_ACTION_NAMES (1) All actions created for Loop In Alarm when the parameter is set in the configuration as a string • SINGLE_ALARM_CLASS_IDS (2) All message class IDs created • SINGLE_ALARM_INFO_TEXTS (3) All info texts created • SINGLE_ALARM_MESSAGE_NUMBERS (4) All message numbers created • SINGLE_ALARM_MESSAGE_TYPE_IDS (5) All message type IDs created • SINGLE_ALARM_GROUP_MESSAGE_CLASSES (6) All message classes created • SINGLE_ALARM_GROUP_MESSAGE_USER_DEFINED (7) All group messages created
pListArray (Variant)	List with the requested content.
Filter (String)	Filters can be set optionally. Wildcards "*" and "?" are also possible.

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 2483)

CommitSingleAlarm Function (Page 2482)

CloseSingleAlarm Function (Page 2481)

VBA in Alarm Logging (Page 2478)

Index

, 54

Adding, 814, 815
DeactivateRTProject, 1464
InquireLanguage, 1466
TlgTrendWindowPressReportSaveButton , 907

A

abort, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
abs, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
AcknowledgeMessage, 836, 838
acos, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

Action, 23, 38, 64, 66, 69, 74, 76, 1644
  Cause for non-execution of an action, 802
  Compiling, 802, 810
  Configuring with VBA, 1644
  Create, 55
  Creating functions and actions, 763, 802
  Creating new actions, 806
  CrossReference, 55
  Defining, 763
  Deleting, 785
  Dialog "Info", 66
  Differences between actions and functions, 802
  edit, 802
  Edit, 55
  Editing, 807, 829
  Exporting, 821
  Finding, 802
  Importing, 821
  Printing, 788
  Protecting with a Password, 66
  Protection against read and write access, 809
  Renaming, 802, 822
  Runtime behavior, 824
  Saving, 810
  Structuring, 763
  Use of DLLs, 774
  Using actions from outside the project, 823
Action configuring, 1644
  with VBA, 1644
Action icon
  Features, 802
Action:Add Information, 64
Action:Configure a Trigger of Type "Tag", 76
Action:Configure a Trigger of Type "Timer", 74
Action:Corrective measure, 67
Action:Delete, 38
Action:Dialog "Info", 64
Action:Editing, 61
Action:new creation, 59
Action:Save, 67
Action:Trigger, 21
Activating, 82
Activating:of global actions in Runtime, 82
ActiveX controls, 1617
  Inserting with VBA, 1617

AddIn, 2414
AddIn Manager, 2415
create (example), 2418
create with VB (example), 2419
load automatically, 2415
load manually, 2415
Loading (instructions), 2417
Loading behavior, 2415
unload, 2415
Use in Graphics Designer, 2414
Adding Timer trigger , 814
adding to picture, 85
Adding trigger, 815
Aktion:Renaming, 81
Aktion:use multiple times, 23
Alarm Control, 265
Alarm Logging, 2478
Creating message with VBA, 2478
Deleting message with VBA, 2478
Editing message with VBA, 2478
Modifying message with VBA, 2478
Alarm object, 115
AlarmControl, 236
AlarmLogs Object, 118
Alarms object (list), 116
Analog Clock, 239
Animation trigger, 73
Application-specific menu, 1573
Add menu entry, 1577
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1575
creating multiple languages, 1580
Application-specific toolbar, 1573
Add icon, 1584
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1582
asctime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
asin, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
atan, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
atan2, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
atof, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
atoi, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
atol, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
AttachDB, 643
Attribute:GSC Runtime, 89
Authorization
Assigning, 820
Authorizations, 820
AXC_OnBtnAlarmHidingList, 858

AXC_OnBtnArcLong, 859
AXC_OnBtnArcShort, 860
AXC_OnBtnComment, 861
AXC_OnBtnEmergAckn, 862
AXC_OnBtnHideDlg, 863
AXC_OnBtnHideUnhideMsg, 864
AXC_OnBtnHit, 865
AXC_OnBtnHornAckn, 866
AXC_OnBtnInfo, 867
AXC_OnBtnLock, 868
AXC_OnBtnLockUnlock, 869
AXC_OnBtnLockWin, 870
AXC_OnBtnLoop, 871
AXC_OnBtnMsgFirst, 872
AXC_OnBtnMsgLast, 873
AXC_OnBtnMsgNext, 874
AXC_OnBtnMsgPrev, 875
AXC_OnBtnMsgWin, 876
AXC_OnBtnPrint, 877
AXC_OnBtnProtocol, 878
AXC_OnBtnScroll, 878
AXC_OnBtnSelect, 879
AXC_OnBtnSinglAckn, 880
AXC_OnBtnSortDlg, 881
AXC_OnBtnTimeBase, 882
AXC_OnBtnVisibleAckn, 883
AXC_SetFilter, 837

B

basics, 760
Basics
    from VBS, 760
Bookmark:delete in Debugger, 105
Bookmark:set in Debugger, 105
Bookmark:skip to, 105
Bookmarks, 105
Breakpoint, 103, 104
Breakpoint:Deleting, 104
Breakpoint:set in Debugger, 103
bsearch, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

C

C action, 1648
    Configuring on an event with VBA, 1648
C script, 1639
    Dynamize property with VBA, 1639
c_bib, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
CalculateStatistic, 643
calloc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ceil, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Change, 78
Change:Triggers, 78
char_io, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Class, 2424
    HMIGO, 2424
clearerr, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

clock, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Code, 45, 108
Code Templates, 61
Code templates for VBS, 45
Code:of a Procedure, 45
Code:print, 108
Collections, 1664
  Overview, 1664
Color code, 778
ComboBox object, 1834
CommonVBSEventArea properties, 2078
CommonVBSPropertyArea property, 2079
Component library, 1593
  Access with VBA, 1593
  Copying an object with VBA, 1596
  Creating a folder with VBA, 1596
  Deleting a folder with VBA, 1596
  Editing with VBA, 1596
  Paste Object into a Picture with VBA, 1599
Computer properties, 91, 771
  Runtime Tab, 91
Configuring, 1570, 1645, 1648
  C action with VBA, 1648
  Direct connection, 1645
  for multiple languages with VBA, 1570
  Trigger with VBA, 1652
  VB action with VBA, 1650
Contents, 13
Controls
  WinCC Alarm Control, 265
  WinCC MediaControl, 246
Controls:HMI Symbol Library 1.4.1, 233
Controls:WinCC Digital Analog Clock, 239
Controls:WinCC Function Trend Control, 267
Controls:WinCC FunctionTrendControl, 241
Controls:WinCC Online Table Control, 269
Controls:WinCC Online Trend Control, 271
Controls:WinCC OnlineTableControl, 247
Controls:WinCC OnlineTrendControl, 250
Controls:WinCC RulerControl, 257
Controls:WinCC Slider Control,WinCC:WinCC Slider Control, 259
Controls:WinCC UserArchiveControl, 263
CopyRows, 643
cos, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
cosh, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Create, 55, 59
  Action, 55
  Procedures, 39
Create:Action, 59
Create:Procedure, 42
CreateTagSet, 644
Cross-reference, 69
ctime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ctype, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Customized Object, 1626
  Access with VBA, 1626
  Deleting, 1628
  Editing with VBA, 1628
  Properties, 1626
  ungroup, 1628
CutRows, 645
Cyclic, 69

D

DataItem Object, 119
DataLogs-Objekt, 121

DataSet-Objekt (List), 122
Debugger, 91, 93, 97, 99, 100, 102, 103, 104, 105, 106, 107
  Activating in WinCC, 91
  Opening after error message, 91
  Opening automatically, 91
  Starting for Global Script, 91
  Starting for Graphics Runtime, 91
Debugger,Microsoft Script Debugger:Components, 95
Debugger:Basic Principles, 93, 102
Debugger:Call Stack, 95
Debugger:Change Picture During Debug, 93
Debugger:Command Window, 95
Debugger:Components, 95
Debugger:Delete bookmark, 105
Debugger:Deleting Breakpoints, 104
Debugger:Determine property values, 106
Debugger:Determine tag values, 106
Debugger:Executing Script Commands, 107
Debugger:modify property values, 106
Debugger:Modify tag values, 106
Debugger:Name of actions in the script file, 99
Debugger:Processing Scripts Step-by-Step, 102
Debugger:Running Documents, 95
Debugger:select running script, 100
Debugger:Set bookmark, 105
Debugger:Setting Breakpoints, 103
Debugger:skip to bookmark, 105
Debugger:Structure of the Script Files, 97
Declaration area, 61
Declaration Area:of an Action, 61
Delete:Actions and Procedures, 38
Delete:Trigger, 79
DeleteRows, 646
DeleteTextLanguage, 2470
Design of Global Script, 30
Design tool, 763
DetachDB, 646
Diagnostics, 84, 85, 86, 87, 88, 89, 90
Diagnostics:Attributes:GSC Diagnostics, 86
Diagnostics:Debugger, 90
Diagnostics:GSC Diagnostics, 84
Diagnostics:GSC Runtime, 87
Diagnostics:GSC Runtime Attributes, 89
Diagnostics:Inserting GSC Diagnostics Window into a Picture, 85
Diagnostics:Inserting GSC Runtime into a Picture, 88
Diagnostics:VBS:Diagnostics, 83
difftime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Digital Analog Clock, 239
Digital Clock, 239
Direct connection, 1645
  Configuring with VBA, 1645
Directio, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
div, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Dynamic dialog, 1636
  Dynamize property with VBA, 1636
Dynamization, 1632
  Dynamic dialog with VBA, 1632
  of properties with VBA, 1632
  Scripts with VBA, 1632
  Tag connection with VBA, 1632
Dynamizing, 1634, 1639
  Property with C script with VBA, 1639
  Property with Dynamic dialog with VBA, 1636
  Property with Tag Connection, 1634
  Property with VB script with VBA, 1642

E

Edit, 55, 647, 1603
  Action, 55
  Copy of a Picture with VBA, 1605
  Layer with VBA, 1603
  Procedures, 39
Edit window
  Working in the edit window, 778
Edit:Action, 61
Editing, 61

Editing function, 780
Editing functions with the keyboard, 780
Editing functions with the mouse, 780
Editing Objects with VBA, 1606
Editing Window, 32
Editing window:Color coding, 32
Editing Window:Global Script, 32
Editor, 28, 30, 36
Editor:Global Script, 28
Editor:Toolbars, 36
Editor:VBS-Editor in the Graphics Designer, 28
EigenschaftenProperties , 1664
Error, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Error codes, 407
Error Messages, 363, 732
Error types, 93
Error types:Logical error, 93
Error types:Runtime Error, 93
Error types:Syntax Error, 93
Error:syntax, 67
Event, 1655
application-specific, 1655
Forwarding, 1655
picture-specific, 1655
Event Handling, 1566, 1655, 1664
Activated event, 1671
Forwarding of events, 1562
Overview, 1664
prevent, 1655
Shutting down, 1655
Event-driven, 69
Event-HandlingVBAEvents , 1664
Events, 1664
Activated, 1671
BeforeClose, 1672
BeforeDocumentClose, 1673
BeforeDocumentSave, 1674
BeforeHMIObjectDelete, 1675
BeforeLibraryFolderDelete, 1676
BeforeLibraryObjectDelete, 1677
BeforeQuit, 1678
BeforeSave, 1679
BeforeVisibleFalse, 1680
ConnectionEvent, 1681
DataLanguageChanged, 1681
DesktopLanguageChanged, 1682
DocumentActivated, 1683
DocumentCreated, 1684
DocumentOpened, 1685
DocumentPropertyChanged, 1687
DocumentSaved, 1686
HMIObjectAdded, 1688
HMIObjectMoved, 1689
HMIObjectPropertyChanged, 1690
HMIObjectResized, 1691
LibraryFolderRenamed, 1692
LibraryObjectAdded, 1694
LibraryObjectRenamed, 1693
MenuItemClicked, 1695
NewLibraryFolder, 1696
NewLibraryObject, 1697
Opened, 1698
Saved, 1699
see reference:Events, 1664
SelectionChanged, 1699
Started, 1700
ToolbarItemClicked, 1701
VBA Event Handling, 1664
ViewCreated, 1702
WindowStateChange, 1704
Example for VBS, 734
Example in WinCC:Configuring a diagnostics output via Trace, 738
Example in WinCC:Configuring a language change, 736
Example in WinCC:Configuring a picture change globally, 737
Example in WinCC:Configuring a picture change via Property, 738
Example in WinCC:Defining the color of objects, 736
Examples, 734

Examples in WinCC, 735, 736, 737, 738, 739, 741, 743, 746
VBS in WinCC, 734
Writing object properties, 743
Examples in WinCC:Accessing objects in Graphics Designer, 735
Examples in WinCC:Deactivate Runtime, 737
Examples in WinCC:Read tag values, 741
Examples in WinCC:Start an Action on the Server (Logging Object), 746
Executing, 1566
VBA Macros, 1566
exit, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ExitWinCC, 1465
exp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Export, 647
External application, 1658
Access with VBA, 1658
Accessing MS Excel with VBA, 1659
log on, 1658

F

fabs, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
FaceplateObject object, 1865
FaceplateType property, 2104
fclose, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Features, 802
LockStatus, 2201
LockText, 2202
LockTextColor, 2203
LongStrokesBold, 2203
LongStrokesOnly, 2204
LongStrokesSize, 2205
LongStrokesTextEach, 2205
Marker, 2207
Max, 2207
MaximizeButton, 2208
MaxZoom, 2208
MCGUBackColorOff, 2209
MCGUBackColorOn, 2210
MCGUBackFlash, 2211
MCGUTextColorOff, 2211
MCGUTextColorOn, 2212
MCGUTextFlash, 2213
MCKOBackColorOff, 2213
MCKOBackColorOn, 2214
MCKOBackFlash, 2215
MCKOTextColorOff, 2215
MCKOTextColorOn, 2216
MCKOTextFlash, 2217
MCKQBackColorOff, 2218
MCKQBackColorOn, 2218
MCKQBackFlash, 2219
MCKQTextColorOff, 2220
MCKQTextColorOn, 2220
MCKQTextFlash, 2221
MCText, 2222
MenuItems, 2222
MenuItemType, 2223
MessageClass, 2224
Min, 2225
MinZoom, 2226
Modified, 2227
Moveable, 2227
Name, 2228
NegativeValue, 2229
Number, 2230
ObjectName, 2232
ObjectSizeDecluttering, 2234
OffsetLeft, 2235
feof, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,

975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ferror, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fflush, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fgetc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fgetpos, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fgets, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
File, 787, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
  Searching, 787
File_pos, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
FillDiagnoseInTags, 1468
floor, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fmod, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Font style
  Setting, 784
fopen, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Forwarding, 1655
  of events, 1655
fputc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fputs, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,

985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fread, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
free, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
freopen, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
frexp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fseek, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
fsetpos, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ftell, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Function
- Application, 790
- Compiling, 786, 790, 799
- Creating, 763, 790
- Creating new function, 793
- Defining, 763
- Differences between actions and functions, 802
- Editing, 790
- Finding, 790
- Functions from other sources, 801
- Printing, 788
- Protection against read and write access, 798
- Renaming, 800
- Saving, 799
- Structuring, 763
- Use of DLLs, 774
- Using internal functions, 795

Function code, 794
- Writing, 794

Function Trend Control, 267

Functions, 2435
CloseSingleAlarm (VBA), 2481
CloseTag (VBA), 2429
CloseTlgArchive (VBA), 2442
CloseTlgTag (VBA), 2443
CommitSingleAlarm (VBA), 2482
CommitTag (VBA), 2430
CommitTlgArchive (VBA), 2445
CommitTlgTag (VBA), 2446
CreateSingleAlarm (VBA), 2483
CreateTag (VBA), 2432
CreateText (VBA), 2467
CreateTextLanguage (VBA), 2466
CreateTlgArchive (VBA), 2448
CreateTlgTag (VBA), 2451
DeleteSingleAlarm (VBA), 2487
DeleteTag (VBA), 2434
DeleteText (VBA), 2468
DeleteTlgArchive (VBA), 2455
DeleteTlgTag (VBA), 2457
GetSingleAlarm (VBA), 2488
GetTag (VBA), 2435
GetText (VBA), 2471
GetTextID (VBA), 2473
GetTlgArchive (VBA), 2458
GetTlgTag (VBA), 2459
ListSingleAlarm (VBA), 2489
ListTag (VBA), 2436
ListText (VBA), 2475
ListTlgArchive (VBA), 2461
ListTlgTag (VBA), 2463
ModifyText (VBA), 2476
FunctionTrendControl, 241
fwrite, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

G

Gauge Control, 243
GCreateMyOperationMsg, 838
GCS-Runtime, 87
GCS-Runtime:Attribute, 89
General Examples, 755, 756, 758, 760
General examples:Configuring a database connection with VBS, 756
General examples:Retrieve MS automation interfaces, 758
General examples:Starting external application, 760
Get_Focus, 1039
GetActualPointLeft, 1048
GetActualPointTop, 1048
GetAdaptBorder, 1105
GetAdaptPicture, 1106
GetAdaptSize, 1107
GetAlarmHigh, 1073
GetAlarmLow, 1074
GetAlignment, 1004
GetAlignmentLeft, 1040
GetAlignmentTop, 1041
GetAssignments, 1062
GetAssumeOnExit, 1063
GetAssumeOnFull, 1064
GetAverage, 1108
GetAxisSection, 1005
GetBackBorderWidth, 1160
GetBackColor, 1013
GetBackColor2, 1013
GetBackColor3, 1014
GetBackColorBottom, 1015
GetBackColorTop, 1015
GetBackFlashColorOff, 1031
GetBackFlashColorOn, 1031
GetBasePicReferenced, 1152
GetBasePicTransColor, 1153
GetBasePicture, 1154
GetBasePicUseTransColor, 1154
GetBitNumber, 1064
GetBorderBackColor, 1016
GetBorderColor, 1017
GetBorderColorBottom, 1017
GetBorderColorTop, 1018
GetBorderEndStyle, 1160
GetBorderFlashColorOff, 1032
GetBorderFlashColorOn, 1033
GetBorderStyle, 1161
GetBorderWidth, 1162
GetBoxAlignment, 1162
GetBoxCount, 1049
GetBoxType, 1108
GetButtonColor, 1019
GetCaption, 1109
GetCheckAlarmHigh, 1074
GetCheckAlarmLow, 1075
GetCheckLimitHigh4, 1076
GetCheckLimitHigh5, 1077
GetCheckLimitLow4, 1077
GetCheckLimitLow5, 1078

GetCheckToleranceHigh, 1079
GetCheckToleranceLow, 1080
GetCheckWarningHigh, 1081
GetCheckWarningLow, 1081
GetClearOnError, 1065
GetClearOnNew, 1066
GetCloseButton, 1110
GetColorAlarmHigh, 1082
GetColorAlarmLow, 1083
GetColorBottom, 1019
GetColorChangeType, 1111
GetColorLimitHigh4, 1083
GetColorLimitHigh5, 1084
GetColorLimitLow4, 1085
GetColorLimitLow5, 1085
GetColorToleranceHigh, 1086
GetColorToleranceLow, 1087
GetColorTop, 1020
GetColorWarningHigh, 1087
GetColorWarningLow, 1088
GetColumn, 648
GetColumnCollection, 649
GetCursorControl, 1111
GetCursorMode, 1112
GetDataFormat, 1067
GetDirection, 1049
GetEditAtOnce, 1113
GetEndAngle, 1050
getenv, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
GetExponent, 1006
GetExtendedOperation, 1114
GetFillColor, 1021
GetFilling, 1029
GetFillingIndex, 1030
GetFillStyle, 1163
GetFillStyle2, 1164
GetFlashBackColor, 1033
GetFlashBorderColor, 1034
GetFlashFlashPicture, 1155
GetFlashForeColor, 1035
GetFlashPicReferenced, 1155
GetFlashPicTransColor, 1156
GetFlashPicture, 1157
GetFlashPicUseTransColor, 1157
GetFlashRateBackColor, 1036
GetFlashRateBorderColor, 1036
GetFlashRateFlashPic, 1158
GetFlashRateForeColor, 1037
GetFontBold, 1041
GetFontItalic, 1042
GetFontName, 1043
GetFontSize, 1044
GetFontUnderline, 1044
GetForeColor, 1022
GetForeFlashColorOff, 1038
GetForeFlashColorOn, 1038
GetGrid, 1051
GetGridColor, 1022
GetGridHeight, 1051
GetGridWidth, 1052
GetHeight, 1053
GetHiddenInput, 1067
GetHitlistColumn , 650
GetHitlisteColumnCollection, 651
GetHotkey, 1115
GetHWDiag, 850
GetHWDiagLevel, 852
GetHysteresis, 1115
GetHysteresisRange, 1116
GetIndex, 1159
GetInputValueChar, 1068
GetInputValueDouble, 1069
GetItemBorderBackColor, 1023
GetItemBorderColor, 1024
GetItemBorderStyle, 1165
GetItemBorderWidth, 1165
GetKopFupAwl, 853
GetKopFupAwlLevel, 854
GetLanguage, 1466
GetLanguageSwitch, 1116
GetLastChange, 1117
GetLayer, 1047
GetLeft, 1053
GetLeftComma, 1006
GetLimitHigh4, 1089
GetLimitHigh5, 1089
GetLimitLow4, 1090
GetLimitLow5, 1091
GetLimitMax, 1091
GetLimitMin, 1092
GetLink, 1104
GetLinkedVariable, 842
GetListType, 1069
GetLocalPicture, 842
GetLongStrokesBold, 1007
GetLongStrokesOnly, 1008
GetLongStrokesSize, 1008

GetLongStrokesTextEach, 1009
GetMarker, 1093
GetMax, 1118
GetMaximizeButton, 1118
GetMessageBlock , 652
GetMessageBlockCollection , 653
GetMessageColumn, 654
GetMessageColumnCollection , 655
GetMin, 1119
GetMoveable, 1120
GetNumberLines, 1070
GetOffsetLeft, 1121
GetOffsetTop, 1121
GetOnTop, 1122
GetOperation, 1122
GetOperationMessage, 1123
GetOperationReport, 1124
GetOperatorMessage , 656
GetOperatorMessageCollection , 657
GetOrientation, 1045
GetOutputFormat, 1071
GetOutputValueChar, 1071
GetOutputValueDouble, 1072
GetParentPicture, 844
GetParentPictureWindow, 844
GetPasswordLevel, 1125
GetPicDeactReferenced, 1137
GetPicDeactTransparent, 1138
GetPicDeactUseTransColor, 1139
GetPicDownReferenced, 1140
GetPicDownTransparent, 1140
GetPicDownUseTransColor, 1141
GetPicReferenced, 1142
GetPicTransColor, 1143
GetPictureDeactivated, 1143
GetPictureDown, 1144
GetPictureName, 1126
GetPictureUp, 1145
GetPicUpReferenced, 1146
GetPicUpTransparent, 1146
GetPicUpUseTransColor, 1147
GetPicUseTransColor, 1148
GetPointCount, 1054
GetPosition, 1135
GetPressed, 1166
GetProcess, 1127
GetPropBOOL, 1149
GetPropChar, 1150
GetPropDouble, 1150
GetPropWord, 1151
GetRadius, 1054
GetRadiusHeight, 1055
GetRadiusWidth, 1056
GetRangeMax, 1136
GetRangeMin, 1136
GetReferenceRotationLeft, 1056
GetReferenceRotationTop, 1057
GetRightComma, 1010
GetRotationAngle, 1058
GetRoundCornerHeight, 1058
GetRoundCornerWidth, 1059
GetRulerBlock , 658
GetRulerBlockCollection , 659
GetRulerColumn, 660
GetRulerColumnCollection , 661
GetRulerData, 662
GetScaleColor, 1024
GetScaleTicks, 1010
GetScaling, 1011
GetScalingType, 1012
GetScrollBars, 1127
GetSelBGColor, 1025
GetSelTextColor, 1026
GetServerName, 1128
GetServerTagPrefix, 1469
GetSizeable, 1129
GetSmallChange, 1129
GetStartAngle, 1060
GetStatisticAreaColumn, 662
GetStatisticAreaColumnCollection, 663
GetStatisticResultColumn, 664
GetStatisticResultColumnCollection, 665
GetStatusbarElement, 666
GetStatusbarElementCollection, 667
GetTagBit, 1409
GetTagBitState, 1369
GetTagBitStateQC, 1389
GetTagBitStateQCWait, 1377
GetTagBitStateWait, 1359
GetTagBitWait, 1400
GetTagByte, 1410
GetTagByteState, 1370
GetTagByteStateQC, 1390
GetTagByteStateQCWait, 1378
GetTagByteStateWait, 1360
GetTagByteWait, 1401
GetTagChar, 1410
GetTagCharState, 1370
GetTagCharStateQC, 1391
GetTagCharStateQCWait, 1379
GetTagCharStateWait, 1361
GetTagCharWait, 1401
GetTagDouble, 1411
GetTagDoubleState, 1371

GetTagDoubleStateQC, 1392
GetTagDoubleStateQCWait, 1380
GetTagDoubleStateWait, 1362
GetTagDoubleWait, 1402
GetTagDWord, 1411
GetTagDWordState, 1372
GetTagDWordStateQC, 1393
GetTagDWordStateQCWait, 1381
GetTagDWordStateWait, 1362
GetTagDWordWait, 1403
GetTagFloat, 1412
GetTagFloatState, 1373
GetTagFloatStateQC, 1394
GetTagFloatStateQCWait, 1382
GetTagFloatStateWait, 1363
GetTagFloatWait, 1403
GetTagMultiStateQCWait, 1383
GetTagMultiStateWait, 1364
GetTagMultiWait, 1404
GetTagPrefix, 1130
GetTagRaw, 1413
GetTagRawState, 1373
GetTagRawStateQC, 1395
GetTagRawStateQCWait, 1384
GetTagRawStateWait, 1365
GetTagRawWait, 1405
GetTagSByte, 1414
GetTagSByteState, 1374
GetTagSByteStateQC, 1396
GetTagSByteStateQCWait, 1385
GetTagSByteStateWait, 1366
GetTagSByteWait, 1406
GetTagSDWord, 1414
GetTagSDWordState, 1375
GetTagSDWordStateQC, 1397
GetTagSDWordStateQCWait, 1386
GetTagSDWordStateWait, 1367
GetTagSDWordWait, 1406
GetTagSWord, 1415
GetTagSWordState, 1376
GetTagSWordStateQC, 1397
GetTagSWordStateQCWait, 1387
GetTagSWordStateWait, 1367
GetTagSWordWait, 1407
GetTagValue, 1415
GetTagValueStateQC, 1398
GetTagValueStateQCWait, 1387
GetTagValueWait, 1408
GetTagWord, 1416
GetTagWordState, 1377
GetTagWordStateQC, 1399
GetTagWordStateQCWait, 1388
GetTagWordStateWait, 1368
GetTagWordWait, 1409
GetText, 1046
GetTimeAxis, 668
GetTimeAxisCollection, 669
GetTimeColumn, 670
GetTimeColumnCollection, 671
GetToggle, 1167
GetToleranceHigh, 1093
GetToleranceLow, 1094
GetToolbarButton, 672
GetToolbarButtonCollection, 673
GetTop, 1060
GetTrend, 674, 1131
GetTrendCollection, 675
GetTrendColor, 1026
GetTrendWindow, 676
GetTrendWindowCollection, 677
GetTypeAlarmHigh, 1095
GetTypeAlarmLow, 1095
GetTypeLimitHigh4, 1096
GetTypeLimitHigh5, 1097
GetTypeLimitLow4, 1098
GetTypeLimitLow5, 1099
GetTypeToleranceHigh, 1099
GetTypeToleranceLow, 1100
GetTypeWarningHigh, 1101
GetTypeWarningLow, 1102
GetUnselBGColor, 1027
GetUnselTextColor, 1028
GetUpdateCycle, 1131
GetValueAxis, 678
GetValueAxisCollection, 679
GetValueColumn, 680
GetValueColumnCollection, 681
GetVisible, 1132
GetWarningHigh, 1102
GetWarningLow, 1103
GetWidth, 1061
GetWindowBorder, 1133
GetWindowsStyle, 1167
GetXAxis, 682
GetXAxisCollection, 683
GetYAxis, 684
GetYAxisCollection, 685
GetZeroPoint, 1062
GetZeroPointValue, 1134
GetZoom, 1134
Global action, 770
    Application, 770
    Characteristics, 770

Global C tags, 772
Application, 772
Definition, 772
Global script
Definition of global C tags, 772
Use of global C tags, 772
Validity range, 772

Global Script, 775
Action icon, 802
Action structure, 763
Adding action-related information, 807
Adding function-related Information, 797
Assigning authorization, 820
Authorizations, 820
Cause for non-execution of an action, 802
Changing trigger, 817
Characteristics of global actions, 770
Characteristics of internal functions, 768
Characteristics of local actions, 769
Characteristics of project functions, 766
Characteristics of standard functions, 767
Color code, 778
Compiling action, 802, 810
Compiling function, 786, 790, 799
Computer properties, 771
Creating action, 763, 802
Creating function, 763
Creating internal function, 793
Creating new actions, 806
Creating new function, 793
Defining action, 763
Defining function, 763
Deleting action, 785
Deleting project functions, 785
Deleting standard functions, 785
Deleting trigger, 818
Design tool, 763
Differences between actions and functions, 802
Editing action, 802, 807, 829
Editing function, 790
Editing functions with the keyboard, 780
Editing functions with the mouse, 780
Effect of triggers on actions, 811
Exporting action, 821
Features, 802
Finding actions, 802
Finding functions, 790
Function structure, 763
Generating new header, 786
GSC Diagnose toolbar,
GSC Runtime, 825
GSC Runtime attributes, 829
Identification of actions in the navigation window, 811
Import action, 821
Including Global Script Runtime in the startup list of the project , 771
Including GSC Runtime in a process picture, 828
Inserting GSC Diagnose in a process picture, 831
New trigger of type Trigger:new trigger of type

Global Script:Adding trigger, 814
New trigger of type Trigger:New trigger of type TagGlobalScript:Adding trigger , 815
Opening, 821
Opening page view, 789
Password input, 798, 809
Printing action, 788
Printing function, 788
Printing project documentation, 789
Properties, 797, 798, 807, 809, 817, 818
Protecting function against read and write access, 798
Protection against read and write access, 809
Renaming action, 802, 822
Renaming function, 800
Runtime behavior of actions, 824
Saving action, 810
Saving file as, 821
Saving function, 799
Searching files, 787
Setting different views, 784
Setting print parameters, 788
Setting the font style, 784
Structure of Global Script editor, 775
System behavior, 802
Trigger type, 763
Use of DLLs in actions, 774
Use of DLLs in functions, 774
Use of global actions, 770
Use of Internal Functions, 768
Use of local actions, 769
Use of project functions, 766
Use of standard functions, 767
Using "Save As...", 784
Using actions from other sources, 823
Using functions, 790
Using Functions from other sources, 801
Using internal functions, 795
Using project functions, 796
Using standard functions, 796
WinCC coding rules, 805
Working in the edit window, 778
Working with toolbars, 781
Writing function code, 794
Global Script editor, 30, 775
Structure, 775
Global Script Runtime, 771
Adding to startup list of the project, 771
Global Script:Deleting Actions and Procedures, 38
Global Script:Toolbars, 36
Global Script:Working in the Editing Window, 32
Global Tag, 27
Global tag:Use in VBS, 27
GMsgFunction, 840
gmtime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Graphic object, 152
Graphic Object:Properties, 152
Graphic Object:Types in VBS, 152
Graphic object:WinCC Gauge Control, 243
Graphics Designer, 1569
@GLOBAL.PDT, 1562
@PROJECT.PDT, 1562
Access to objects, 1606
Access to the component library with VBA, 1593
Access to the symbol library with VBA, 1593
Adapting with VBA, 1567, 1569
Add menu entry to user-defined menu, 1577
Adding a New Icon to the used-defined Toolbar, 1584
Application-specific menu, 1573
Application-specific toolbar, 1573
Assigning VBA macro to user-defined menu, 1589
Assigning VBA macro to user-defined toolbar, 1589
Configure user-defined menu, 1572
Configure user-defined toolbar, 1572
Create an application-specific menu, 1575
Create toolbar, 1582
Creating Menus in Multiple Languages, 1580
Editing objects with VBA, 1606
Editing Objects with VBA, 1606
Executing VBA macros, 1566
Exporting VBA code, 1565
Importing VBA code, 1565
Object model, 1664
Organizing the VBA code, 1562
Paste Object from the component library into a picture with VBA, 1599
Paste Object from the symbol library into a picture with VBA, 1599
picture-specific menu, 1573
picture-specific toolbar, 1573
Template file, 1562
User-defined menu, 1573
User-defined toolbar, 1573

Group Objects, 1618
Access to object properties, 1618
Adding an object with VBA, 1620
basics, 1618
Creating with VBA, 1620
Deleting with VBA, 1620
Editing existing objects, 1623
Editing with VBA, 1618, 1620
Removing an object with VBA, 1620
Ungrouping with VBA, 1620
Grouping, 1618
Access to object properties, 1618
Adding an object with VBA, 1620
basics, 1618
Creating with VBA, 1620
Deleting with VBA, 1620
Editing existing objects, 1623
Editing with VBA, 1618, 1620
Removing an object with VBA, 1620
Ungrouping with VBA, 1620
GSC Diagnose
Including in process picture, 831
Toolbar,
GSC Diagnostics, 86
GSC Diagnostics Attributes, 86
GSC Diagnostics: inserting into a Picture, 88
GSC Diagnostics:Attributes, 86
GSC Diagnostics:Toolbar, 86
GSC Runtime
Attributes, 829
Including in process picture, 828
GSC Runtime:Inserting into a Picture, 88

H

Header, 786
Regenerating, 786
HideAlarm, 686
Highlight Syntax, 45, 61
HMIGO class, 2424
Error Handling, 2424
Usage, 2424
HMIObjects, 1664
see HMIObjects , 1664
see lists, 1664
HMIRuntime Object, 125

I

Information, 49, 64
Adding action-related information, 807
Information:add to action, 64
Information:add to module, 49
Information:add to procedure, 49
Insert:GSC Diagnostics, 85
Insert:GSC Runtime, 88
Intellisense, 45, 61
Internal function, 768
Application, 768
Characteristics, 768
isahum, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isalpha, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isdigit, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isgraph, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
islower, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isprint, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,

952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ispunct, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isspace, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isupper, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
isxdigit, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Item Object, 126

L

labs, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Language, 406
Layer, 1603
  Controlling the visibility with VBA, 1603
  CS, 1603
  Editing with VBA, 1603
  RT, 1603
Layer Object, 127
Layers Object (Listing), 129
ldexp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ldiv, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ListBox object, 1900
Lists
  Actions, 1810
  AnalogResultInfo, 1813
  Application, 1814
  DataLanguages, 1842
  Documents, 1850
  Events, 1863
  FolderItems, 1868
  GroupedObjects, 1877
  HMIDefaultObjects, 1878
  HMIObjects, 1884
  LanguageFonts, 1890
  LanguageTexts, 1893
  Layers, 1895
  MenuItems, 1909
  Menus, 1904
  Overview, 1664
  Properties, 1931
  QualityCodeStateValues, 1936
  Selection, 1949
  SymbolLibraries, 1963
  ToolbarItems, 1973
  Toolbars, 1968
  VariableStateValues, 1987
  VariableTriggers, 1990
  Views, 1993

Lists in VBS
Alarms object (list), 116
DataSet Object (List), 122
Layers Object (Listing), 129
ProcessValues Object (Lists), 132
ScreenItems object (list), 138
Screens Object (Lists), 143
Tags Object (List), 149
TagSet Object (List), 151
Local actions, 769
Application, 769
Characteristics, 769
localtime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
LockAlarm, 688
log, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
log10, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Logging Object, 130
Logical errors, 93
LoopInAlarm, 688
lpszPictureName, 835

M

malloc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
math, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
MediaControl, 246
memchr, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
memcmp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
memcpy, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
memmove, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
memory, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
memset, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,

952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Menu, 1573, 1601
Add menu entry, 1577
application-specific, 1573
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1575
creating multiple languages, 1580
picture-specific, 1573, 1601
Placement, 1573
Properties, 1573
User-defined, 1573
Menu bar, 775
Menus and toolbars
VBScript, 14
Methods, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 702, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 725, 729, 730, 731, 732, 1664
Activate, 1704
Add, 1705
Add (AnalogResultInfos Listing), 1706
Add (CustomToolbars Listing), 1708
Add (Documents Listing), 1709
Add (GroupedObjects), 1710
Add (TagTriggers Listing), 1712
Add (Views Listing), 1713
AddAction, 1713
AddActiveXControl, 1714
AddFolder, 1716
AddFromClipboard, 1717
AddHMIObject, 1719
AddItem, 1720
AddOLEObject, 1722
AlignBottom, 1724
AlignLeft, 1725
AlignRight, 1726
AlignTop, 1727
ArrangeMinimizedWindows, 1728
BackwardOneLevel, 1729
BringToFront, 1730
CascadeWindows, 1731
CenterHorizontally, 1732
CenterVertically, 1733
CheckSyntax, 1734
Close, 1735
CloseAll, 1736
ConvertToScript, 1737
CopySelection, 1738
CopyToClipboard, 1740
CreateCustomizedObject, 1741
CreateDynamic, 1742
CreateGroup, 1743
Delete, 1745
DeleteAll, 1746
DeleteDynamic, 1747
Destroy, 1749
DuplicateSelection, 1750
EvenlySpaceHorizontally, 1751
EvenlySpaceVertically, 1752
Export, 1754
Find, 1755
FindByDisplayName, 1756
FlipHorizontally, 1757
FlipVertically, 1758
ForwardOneLevel, 1759
GetDeclutterObjectSize, 1744
GetItemByPath, 1760
InsertFromMenuItem, 1762
InsertMenu, 1763
InsertMenuItem, 1765
InsertSeparator, 1766
InsertSubmenu, 1767
InsertToolbarItem, 1769
IsCSLayerVisible, 1770
IsRTLayerVisible, 1771
Item, 687, 1772
ItemByLcid, 1774

Methods in VBS
ActivateDynamic, 640
Create, 644
DeactivateDynamic, 646
Methods:Activate, 639
Methods:Add, 641
Methods:Read, 698
Methods:Refresh, 702
Methods:Remove, 703
Methods:RemoveAll, 707
Methods:Restore, 708
Methods:Stop, 723
Methods:Trace, 724
Methods:Write, 725
Microsoft Script Debugger, 90, 95
mktime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
modf, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Module, 39, 42
Dialog "Info", 51
Editing, 39
Memory, 52
Name, 39
Protecting with a Password, 51
Module:Add Information, 49
Module:Create, 42
Module:Dialog "Info", 49
Module:Renaming, 54
MoveAxis, 689
MoveToFirst, 689
MoveToFirstLine, 689
MoveToFirstPage, 690
MoveToLast, 690
MoveToLastLine, 691
MoveToLastPage, 691
MoveToNext, 691
MoveToNextLine, 692
MoveToNextPage, 692
MoveToPrevious, 693
MoveToPreviousLine, 693
MoveToPreviousPage, 693
MultiLineEdit object, 1910

N

Names, 99
Names:of actions in VBScript Files, 99
Navigation window
Identification of actions in the navigation window, 811
New, 42, 59
New:Action, 59
New:Procedure, 42
NextColumn, 694
NextTrend, 694

O

Object model, 1664
Lists, 113
Methods, 638
Objects, 113
Object model:Properties, 276
Object names, 99
Object Types
Controls, 230
Customized object, 273
Object Types:Group, 274

Objects, 1664
3DBarGraph, 1805
ActiveXControl, 1811
AnalogResultInfo, 1812
ApplicationWindow, 1817
BarGraph, 1819
BinaryResultInfo, 1822
BitResultInfo, 1823
Button, 1824
CheckBox, 1827
Circle, 1828
CircularArc, 1831
ConnectorPoints (Listing), 1837
CustomizedObject, 1839
DataLanguage, 1841
DestLink, 1843
DirectConnection, 1845
Document, 1847
DynamicDialog, 1851
Ellipse, 1853
EllipseArc, 1856
EllipseSegment, 1859
Event, 1862
FolderItem , 1867
GraphicObject, 1870
Group, 1873
GroupDisplay, 1874
HMIObject, 1882
IOField, 1886
LanguageFont, 1889
LanguageText , 1892
Layer, 1894
Line, 1897
Menu, 1903
MenuItem , 1906
objConnection, 1912
OLEObject, 1914
OptionGroup, 1916
Overview, 1664
PictureWindow, 1919
PieSegment, 1922
Polygon, 1925
PolyLine, 1928
Property, 1932
QualityCodeStateValue, 1934
Rectangle, 1939
RoundButton, 1942
RoundRectangle, 1945
ScriptInfo, 1948
Slider, 1952
SourceLink, 1955
StaticText, 1956
StatusDisplay, 1959
SymbolLibrary, 1962
TextList, 1964
Toolbar, 1967
ToolbarItem , 1970
Trigger, 1974
VariableStateValue, 1986
VariableTrigger , 1989
View, 1991
Objects in VBA
ComboBox , 1834
FaceplateObject, 1865
ListBox, 1900
MultiLineEdit, 1910
TubeArcObject, 1976
TubeDoubleTeeObject, 1979
TubePolyline, 1981
TubeTeeObject, 1984
Objects in VBS
Alarm object, 115
AlarmLogs Object, 118
DataItem Object, 119
DataLogs Object, 121
HMIRuntime Object, 125
Item Object, 126
Layer Object, 127
Logging Object, 130
ProcessValue Object, 131
Project Object, 133
Screen Object, 140
ScreenItem Object, 134
Smart tags, 145
Tag Object, 146
ObjekteVBA-Referenz
ObjekteundAuflistungen , 1664
Objektnames:define in Runtime, 99
OnDeactivateExecute, 855
OnErrorExecute, 856
OneToOneView, 695
Online Trend Control, 271
OnlineTableControl, 247
OnlineTrendControl, 250
OnTime, 857
OpenHomePicture, 1352
Opening, 821
OpenNextPicture, 1352
OpenPicture, 845
OpenPrevPicture, 1353
OpenStoredPicture, 1353
Output, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Output window, 775

P

Page view
Opening, 789
Password, 51, 66
assign fro action, 66
for a module, 51
for a Procedure, 51
Password input, 798, 809
Paste, 85, 88
PasteRows , 695
Picture, 1605
Editing copy with VBA, 1605
Picture-specific menu, 1602
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1602
creating multiple languages, 1580
Picture-specific toolbar, 1602
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1602
pow, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
PreviousColumn, 695
PreviousTrend, 696
Print, 696
Print parameters, 788
Setting, 788
Print:VBScript, 108
procedure, 1566
Creating, 39
Dialog "Info", 51
Editing, 39
Memory, 52
Name, 39
Protecting with a Password, 51
Version, 1566
Procedure:Add Information, 49
Procedure:Change, 45
Procedure:Delete, 38
Procedure:Dialog "Info", 49
Procedure:New creation, 42
Procedure:Project Procedure, 47
Procedure:Rename, 54
Procedure:Standard procedure, 47
Procedure:Storage location in WinCC, 23
Procedure:use multiple times, 23
Procedure:Write Code, 45
ProcessValue Object, 131
ProcessValues Object (List), 132
ProgramExecute, 858
Project documentation
Printing, 789
Project function, 766
Application, 766
Applying, 796
Characteristics, 766
Deleting, 785
Project language, 1570
Access with VBA, 1570
Project Object, 133
Project Procedure:Use, 47

Properties, 797, 798, 807, 809, 817, 818
Actions, 1995
ActiveDocument, 1996
ActiveLayer, 1996
ActualPointLeft, 1997
ActualPointTop, 1998
AdaptBorder, 1999
AdaptPicture, 2000
AdaptSize, 2001
AlarmHigh, 2002
AlarmLow, 2002
Alignment, 2003
AlignmentLeft, 2004
AlignmentTop, 2005
AnalogResultInfos, 2006
AngleAlpha, 2006
AngleBeta, 2007
Application, 2008
ApplicationDataPath, 2008
Assignments, 2009
AssumeOnExit, 2010
AssumeOnFull, 2010
AutomationName, 2011
AvailableDataLanguages, 2013
Average, 2014
Axe, 2014
AxisSection, 2015
BackBorderWidth, 2016
BackColor, 2017
BackColor2, 2018
BackColor3, 2019
BackColorBottom, 2020
BackColorTop, 2020
BackFlashColorOff, 2021
BackFlashColorOn, 2022
Background, 2023
BackPictureAlignment, 2024
BackPictureName, 2025
BarDepth, 2025
BarHeight, 2026
BarWidth, 2026
BasePicReferenced, 2027
BasePicTransColor, 2027
BasePicture, 2028
BasePicUseTransColor, 2029
BaseX, 2030
BaseY, 2030
BinaryResultInfo, 2031
BitNotSetValue, 2031
BitNumber, 2032
BitResultInfo, 2033
BitSetValue, 2033
Bold, 2034
BorderBackColor, 2035
BorderColor, 2036
BorderColorBottom, 2038
BorderColorTop, 2038
BorderEndStyle, 2039
BorderFlashColorOff, 2040
BorderFlashColorOn, 2042
BorderStyle, 2043
Properties in VBA
CommonVBSEventArea, 2078
CommonVBSPropertyArea, 2079
FaceplateType, 2104
TagScaleParam1, 2306
TagScaleParam2, 2306
TagScaleParam3, 2306
TagScaleParam4, 2306
Properties in VBS, 276
AlarmID, 282
BackStyle, 299
Comment, 344
ComputerName, 345
Context, 345
FillStyle, 369
FillStyleAlignment, 371
GlobalColorScheme, 386
HiddenInput, 285, 286, 392, 430, 459, 484, 514, 606, 635
HitlistMaxSourceItems, 395
HitlistMaxSourceItemsWarn, 395
HitlistRelTimeFactor, 396
HitlistRelTimeFactorType, 396
IndependentWindow, 399
Index, 400
InputValue, 402
Instance, 402
ItemProviderClsid, 404
LineJoinStyle, 427
NumberLines, 454
ProcessValue, 485
RotationAngle, 493
SavedTrend, 498
Selected Trend, 505
SelIndex, 508
SelText, 508
Smart tag, 517
SortOrder, 517
State, 524
TableFocusOnButtonCommand, 530
Transparency, 577
UserName, 606
WinCCStyle, 627
WindowMonitorNumber, 628
WindowPositionMode, 628
WindowsStyle, 628
Properties in VBS/ScrollPositionX, 502
Properties in VBS/ScrollPositionY, 502
Properties in VBS: DesiredCurveSourceUAColumnY, 357
Properties in VBS:AccessPath, 276
Properties in VBS:Activate, 277

Properties in VBS:ActiveProject, 278
Properties in VBS:ActiveScreen, 278
Properties in VBS:ActiveScreenItem, 279
Properties in VBS:Actualize, 279
Properties in VBS:ActualPointLeft, 280
Properties in VBS:ActualPointTop, 280
Properties in VBS:AdaptBorder, 280
Properties in VBS:AdaptPicture, 281
Properties in VBS:AdaptSize, 281
Properties in VBS:AdjustRuler, 282
Properties in VBS:AlarmHigh, 282
Properties in VBS:AlarmLogs, 283
Properties in VBS:AlarmLow, 283
Properties in VBS:Alignment, 283
Properties in VBS:AlignmentLeft, 283
Properties in VBS:AlignmentTop, 284
Properties in VBS:AllowPersistence, 285
Properties in VBS:AllServer, 285
Properties in VBS:Analog, 286
Properties in VBS:AngleAlpha, 286
Properties in VBS:AngleBeta, 286
Properties in VBS:AngleMax, 287
Properties in VBS:AngleMin, 287
Properties in VBS:Application, 287
Properties in VBS:Archive, 288
Properties in VBS:Assignments, 289
Properties in VBS:AssumeOnExit, 289
Properties in VBS:AssumeOnFull, 289
Properties in VBS:AutoRange, 291
Properties in VBS:AutoRangeX, 291
Properties in VBS:AutoRangeY, 291
Properties in VBS:AutoScroll, 291
Properties in VBS:AutoSize, 293
Properties in VBS:Average, 294
Properties in VBS:Axe, 294
Properties in VBS:Axis, 537
Properties in VBS:AxisSection, 294
Properties in VBS:BackBorderWidth, 295
Properties in VBS:BackColor, 295
Properties in VBS:BackColor2, 296
Properties in VBS:BackColor3, 296
Properties in VBS:BackColorBottom, 297
Properties in VBS:BackColorTop, 297
Properties in VBS:BackFlashColorOff, 297
Properties in VBS:BackFlashColorOn, 297
Properties in VBS:Background, 298
Properties in VBS:Background Picture, 298
Properties in VBS:BackPictureAlignment, 298
Properties in VBS:BackPictureName, 298
Properties in VBS:BarBackColor, 300
Properties in VBS:BarDepth, 300
Properties in VBS:BarFillColor, 300
Properties in VBS:BarHeight, 300
Properties in VBS:BarWidth, 301
Properties in VBS:BasePicReferenced, 301
Properties in VBS:BasePicTransColor, 301
Properties in VBS:BasePicture, 302
Properties in VBS:BasePicUseTransColor, 302
Properties in VBS:BaseScreenName, 302
Properties in VBS:BaseX, 303
Properties in VBS:BaseY, 303
Properties in VBS:BeginTime, 304
Properties in VBS:BeginValue, 304
Properties in VBS:BeginWidth, 307
Properties in VBS:BeginX, 304
Properties in VBS:BeginY, 305
Properties in VBS:BevelColorDown, 305
Properties in VBS:BevelColorUp, 305
Properties in VBS:BevelInner, 306
Properties in VBS:BevelOuter, 306
Properties in VBS:BitNumber, 307
Properties in VBS:BlinkColor, 307
Properties in VBS:BlinkMode, 307
Properties in VBS:BlinkSpeed, 308
Properties in VBS:BorderBackColor, 312
Properties in VBS:BorderColor, 313
Properties in VBS:BorderColorBottom, 313
Properties in VBS:BorderColorTop, 313
Properties in VBS:BorderEndStyle, 314
Properties in VBS:BorderFlashColorOff, 314
Properties in VBS:BorderFlashColorOn, 314
Properties in VBS:BorderStyle, 314
Properties in VBS:BorderWidth, 315
Properties in VBS:BottomConnectedConnectionPointIndex, 315
Properties in VBS:BottomConnectedObjectName, 316
Properties in VBS:BoxAlignment, 316
Properties in VBS:BoxCount, 316
Properties in VBS:BoxType, 317
Properties in VBS:Button1Width, 318
Properties in VBS:Button2Width, 319
Properties in VBS:Button3Width, 319
Properties in VBS:Button4Width, 319
Properties in VBS:ButtonColor, 317
Properties in VBS:ButtonCommand, 317
Properties in VBS:Caption, 320
Properties in VBS:CaptionColor, 320
Properties in VBS:CaptionFont, 321
Properties in VBS:CaptionOffset, 321
Properties in VBS:CaptionText, 321
Properties in VBS:CellCut, 322
Properties in VBS:CenterColor, 323
Properties in VBS:CenterScale, 323

Properties in VBS:CheckAlarmHigh, 324
Properties in VBS:CheckAlarmLow, 324
Properties in VBS:CheckLimitHigh4, 324
Properties in VBS:CheckLimitHigh5, 325
Properties in VBS:CheckLimitLow4, 325
Properties in VBS:CheckLimitLow5, 325
Properties in VBS:CheckToleranceHigh, 326
Properties in VBS:CheckToleranceLow, 326
Properties in VBS:CheckWarningHigh, 326
Properties in VBS:CheckWarningLow, 327
Properties in VBS:ClearOnError, 327
Properties in VBS:ClearOnNew, 327
Properties in VBS:Closeable, 327
Properties in VBS:CloseButton, 328
Properties in VBS:CoarseGrid, 328
Properties in VBS:CoarseGridValue, 329
Properties in VBS:CoarseGridValueX, 330
Properties in VBS:CoarseGridValueY, 330
Properties in VBS:CoarseGridX, 329
Properties in VBS:CoarseGridY, 329
Properties in VBS:CollectValue, 330
Properties in VBS:ColMove, 331
Properties in VBS:Color, 331
Properties in VBS:ColorAlarmHigh, 331
Properties in VBS:ColorAlarmLow, 332
Properties in VBS:ColorBottom, 332
Properties in VBS:ColorChangeType, 332
Properties in VBS:ColorLimitHigh4, 333
Properties in VBS:ColorLimitHigh5, 333
Properties in VBS:ColorLimitLow4, 333
Properties in VBS:ColorLimitLow5, 334
Properties in VBS:ColorToleranceHigh, 334
Properties in VBS:ColorToleranceLow, 334
Properties in VBS:ColorTop, 335
Properties in VBS:ColorWarningHigh, 335
Properties in VBS:ColorWarningLow, 335
Properties in VBS:ColTitle, 336
Properties in VBS:ColWidth, 343
Properties in VBS:Command, 344
Properties in VBS:CommonTime, 344
Properties in VBS:CommonX, 345
Properties in VBS:CommonY, 345
Properties in VBS:ContinuousChange, 346
Properties in VBS:Count, 346
Properties in VBS:CurrentContext, 347
Properties in VBS:Cursor, 348
Properties in VBS:CursorControl, 348
Properties in VBS:CurveForm, 349
Properties in VBS:Danger, 350
Properties in VBS:DangerColor, 350, 353
Properties in VBS:DataFormat, 350
Properties in VBS:DataIndex, 351
Properties in VBS:DataLogs, 351
Properties in VBS:DataSet, 351
Properties in VBS:DataX, 352
Properties in VBS:DataXY, 352
Properties in VBS:DataY, 352
Properties in VBS:DeleteData, 354
Properties in VBS:Delta, 355
Properties in VBS:DesiredCurveColor, 355
Properties in VBS:DesiredCurveCurveForm, 355
Properties in VBS:DesiredCurveSourceNumberOfUAValues, 356
Properties in VBS:DesiredCurveSourceUAArchive, 356
Properties in VBS:DesiredCurveSourceUAArchiveStartID, 356
Properties in VBS:DesiredCurveSourceUAColumnX, 357
Properties in VBS:DesiredCurveVisible, 357
Properties in VBS:Direction, 357
Properties in VBS:DisplayOptions, 358
Properties in VBS:Edit, 359
Properties in VBS:Editable, 359
Properties in VBS:EditAtOnce, 360
Properties in VBS:Enabled, 360
Properties in VBS:EndAngle, 361
Properties in VBS:EndTime, 362
Properties in VBS:EndValue, 362
Properties in VBS:EndX, 363
Properties in VBS:EndY, 363
Properties in VBS:ErrorDescription, 363
Properties in VBS:Exponent, 364
Properties in VBS:ExtendedOperation, 367
Properties in VBS:ExtendedZoomingEnable, 367
Properties in VBS:FillColor, 368
Properties in VBS:Filling, 368
Properties in VBS:FillingIndex, 369
Properties in VBS:FillStyle2, 370
Properties in VBS:FineGrid, 371
Properties in VBS:FineGridValue, 371
Properties in VBS:FineGridValueX, 372
Properties in VBS:FineGridValueY, 372
Properties in VBS:FineGridX, 372
Properties in VBS:FineGridY, 373
Properties in VBS:FlashBackColor, 373
Properties in VBS:FlashBorderColor, 373
Properties in VBS:FlashFlashPicture, 373
Properties in VBS:FlashForeColor, 374
Properties in VBS:FlashPicReferenced, 374
Properties in VBS:FlashPicTransColor, 374
Properties in VBS:FlashPicture, 375
Properties in VBS:FlashPicUseTransColor, 375
Properties in VBS:FlashRate, 375

Properties in VBS:FlashRateBackColor, 376
Properties in VBS:FlashRateBorderColor, 376
Properties in VBS:FlashRateFlashPic, 376
Properties in VBS:FlashRateForeColor, 377
Properties in VBS:Flip, 377
Properties in VBS:FocusColor, 378
Properties in VBS:FocusRect, 378
Properties in VBS:FocusWidth, 378
Properties in VBS:Font, 379
Properties in VBS:FontBold, 380
Properties in VBS:FontItalic, 380
Properties in VBS:FontName, 381
Properties in VBS:FontPosition, 381
Properties in VBS:FontSize, 382
Properties in VBS:FontStrikeThru, 382
Properties in VBS:FontUnderline, 382
Properties in VBS:ForeColor, 383
Properties in VBS:ForeFlashColorOff, 383
Properties in VBS:ForeFlashColorOn, 384
Properties in VBS:FrameColor, 384
Properties in VBS:FrameColorDown, 385
Properties in VBS:FrameColorUp, 385
Properties in VBS:FramePicture, 385
Properties in VBS:FrameScale, 385
Properties in VBS:FrameWidth, 386
Properties in VBS:FreezeProviderConnections, 386
Properties in VBS:GraphDirection, 387
Properties in VBS:GridLineHorz, 388
Properties in VBS:Gridlines, 387, 388
Properties in VBS:GridLinesValueX, 388
Properties in VBS:GridLinesValueY, 389
Properties in VBS:GridlinesX, 389
Properties in VBS:GridLinesY, 389
Properties in VBS:GridLineValue, 390
Properties in VBS:GridLineVert, 390
Properties in VBS:HandFillColor, 390
Properties in VBS:Handtype, 391
Properties in VBS:HeaderSort, 391
Properties in VBS:Height, 391
Properties in VBS:HiddenInput, 392
Properties in VBS:Hotkey, 397
Properties in VBS:HourNeedleHeight, 398
Properties in VBS:HourNeedleWidth, 398
Properties in VBS:Hysteresis, 399
Properties in VBS:HysteresisRange, 399
Properties in VBS:InnerBevelOffset, 401
Properties in VBS:InnerBevelStyle, 401
Properties in VBS:InnerBevelWidth, 401
Properties in VBS:InsertData, 402
Properties in VBS:ItemBorderBackColor, 403
Properties in VBS:ItemBorderColor, 403
Properties in VBS:ItemBorderStyle, 403
Properties in VBS:ItemBorderWidth, 404
Properties in VBS:ItemVisible, 404
Properties in VBS:Label, 405
Properties in VBS:LabelColor, 405
Properties in VBS:LabelX, 405
Properties in VBS:LabelY, 406
Properties in VBS:Language-IDs, 406
Properties in VBS:LanguageSwitch, 406
Properties in VBS:LastError, 407
Properties in VBS:Layer, 408
Properties in VBS:Layer00Checked, 409
Properties in VBS:Layer00Color, 412
Properties in VBS:Layer00Value, 419
Properties in VBS:Layer01Checked, 409
Properties in VBS:Layer01Color, 413
Properties in VBS:Layer01Value, 419
Properties in VBS:Layer02Checked, 410
Properties in VBS:Layer02Color, 413
Properties in VBS:Layer02Value, 420
Properties in VBS:Layer03Checked, 410
Properties in VBS:Layer03Color, 413
Properties in VBS:Layer03Value, 420
Properties in VBS:Layer04Checked, 410
Properties in VBS:Layer04Color, 413
Properties in VBS:Layer04Value, 420
Properties in VBS:Layer05Checked, 410
Properties in VBS:Layer05Color, 414
Properties in VBS:Layer05Value, 420
Properties in VBS:Layer06Checked, 411
Properties in VBS:Layer06Color, 414
Properties in VBS:Layer06Value, 421
Properties in VBS:Layer07Checked, 411
Properties in VBS:Layer07Color, 414
Properties in VBS:Layer07Value, 421
Properties in VBS:Layer08Checked, 411
Properties in VBS:Layer08Color, 415
Properties in VBS:Layer08Value, 421
Properties in VBS:Layer09Checked, 412
Properties in VBS:Layer09Color, 415
Properties in VBS:Layer09Value, 422
Properties in VBS:Layer10Checked, 412
Properties in VBS:Layer10Color, 415
Properties in VBS:Layer10Value, 422
Properties in VBS:LayerDeclutteringEnable, 422
Properties in VBS:Layers, 423
Properties in VBS:Left, 423
Properties in VBS:LeftComma, 424
Properties in VBS:LightEffect, 424
Properties in VBS:LimitHigh4, 424
Properties in VBS:LimitHigh5, 425
Properties in VBS:LimitLow4, 425
Properties in VBS:LimitLow5, 425

Properties in VBS:LimitMax, 426
Properties in VBS:LimitMin, 426
Properties in VBS:LineFont, 426
Properties in VBS:LineHeight, 427
Properties in VBS:LineTitle, 427
Properties in VBS:LineWidth, 428
Properties in VBS:ListType, 428
Properties in VBS:LoadDataImmediately, 429
Properties in VBS:LocaleID, 429
Properties in VBS:LockBackColor, 430
Properties in VBS:LockStatus, 430
Properties in VBS:LockText, 430
Properties in VBS:LockTextColor, 431
Properties in VBS:Logging, 431
Properties in VBS:LongStrokesBold, 431
Properties in VBS:LongStrokesOnly, 432
Properties in VBS:LongStrokesSize, 432
Properties in VBS:LongStrokesTextEach, 432
Properties in VBS:LowerLimit, 433
Properties in VBS:LowerLimitColor, 434
Properties in VBS:LowerLimitValue, 434
Properties in VBS:Marker, 435
Properties in VBS:Max, 435
Properties in VBS:MaximizeButton, 435
Properties in VBS:MCGUBackColorOff, 436
Properties in VBS:MCGUBackColorOn, 436
Properties in VBS:MCGUBackFlash, 436
Properties in VBS:MCGUTextColorOff, 437
Properties in VBS:MCGUTextColorOn, 437
Properties in VBS:MCGUTextFlash, 437
Properties in VBS:MCKOBackColorOff, 437
Properties in VBS:MCKOBackColorOn, 438
Properties in VBS:MCKOBackFlash, 438
Properties in VBS:MCKOTextColorOff, 438
Properties in VBS:MCKOTextColorOn, 439
Properties in VBS:MCKOTextFlash, 439
Properties in VBS:MCKQBackColorOff, 439
Properties in VBS:MCKQBackColorOn, 439
Properties in VBS:MCKQBackFlash, 440
Properties in VBS:MCKQTextColorOff, 440
Properties in VBS:MCKQTextColorOn, 440
Properties in VBS:MCKQTextFlash, 441
Properties in VBS:MCText, 441
Properties in VBS:MeasurePoints, 441
Properties in VBS:MessageClass, 447
Properties in VBS:Min, 449
Properties in VBS:MinuteNeedleHeight, 450
Properties in VBS:MinuteNeedleWidth, 450
Properties in VBS:Movable, 451
Properties in VBS:MsgCtrlFlags, 451
Properties in VBS:MsgFilterSQL, 452
Properties in VBS:Name, 452
Properties in VBS:NeedleColor, 453
Properties in VBS:NormalColor, 454
Properties in VBS:NumItems, 454
Properties in VBS:Object, 455
Properties in VBS:ObjectName, 455
Properties in VBS:ObjectSizeDeclutteringEnable, 456
Properties in VBS:ObjectSizeDeclutteringMax, 457
Properties in VBS:ObjectSizeDeclutteringMin, 457
Properties in VBS:OffsetLeft, 458
Properties in VBS:OffsetTop, 458
Properties in VBS:Online, 459
Properties in VBS:OnTop, 460
Properties in VBS:OperationMessage, 460
Properties in VBS:OperationReport, 461
Properties in VBS:Orientation, 468
Properties in VBS:OuterBevelStyle, 468
Properties in VBS:OuterBevelWidth, 469
Properties in VBS:Outline, 469
Properties in VBS:OutputFormat, 469
Properties in VBS:OutputValue, 470
Properties in VBS:Parent, 471
Properties in VBS:PasswordLevel, 472
Properties in VBS:Path, 473
Properties in VBS:PersistentRT, 474
Properties in VBS:PersistentRTCS, 474
Properties in VBS:PersistentRTCSPermission, 475
Properties in VBS:PersistentRTPermission, 475
Properties in VBS:PicDeactReferenced, 476
Properties in VBS:PicDeactTransparent, 476
Properties in VBS:PicDeactUseTransColor, 476
Properties in VBS:PicDownReferenced, 477
Properties in VBS:PicDownTransparent, 477
Properties in VBS:PicDownUseTransColor, 477
Properties in VBS:PicReferenced, 477
Properties in VBS:PicTransColor, 478
Properties in VBS:Picture, 478
Properties in VBS:PictureBack, 478
Properties in VBS:PictureDeactivated, 479
Properties in VBS:PictureDown, 479
Properties in VBS:PictureName, 480
Properties in VBS:PictureSelected, 480
Properties in VBS:PictureThumb, 480
Properties in VBS:PictureUnselected, 480
Properties in VBS:PictureUp, 479
Properties in VBS:PicUpReferenced, 481
Properties in VBS:PicUpTransparent, 481
Properties in VBS:PicUpUseTransColor, 481
Properties in VBS:PicUseTransColor, 482
Properties in VBS:PointCount, 482
Properties in VBS:Position, 482
Properties in VBS:Precisions, 483

Properties in VBS:PrecisionX, 483
Properties in VBS:PrecisionY, 483
Properties in VBS:PredefinedAngles, 484
Properties in VBS:Pressed, 484
Properties in VBS:PrintJob, 484
Properties in VBS:Process, 485
Properties in VBS:ProjectPath, 486
Properties in VBS:ProviderClsID, 486
Properties in VBS:ProviderType, 486
Properties in VBS:QualityCode, 487
Properties in VBS:Radius, 488
Properties in VBS:RadiusHeigth, 488
Properties in VBS:RadiusWidth, 488
Properties in VBS:RangeMax, 489
Properties in VBS:RangeMin, 489
Properties in VBS:Rectangular, 489
Properties in VBS:ReferenceRotationLeft, 490
Properties in VBS:ReferenceRotationTop, 490
Properties in VBS:RelayCurves, 490
Properties in VBS:Relevant, 491
Properties in VBS:Replacement, 491
Properties in VBS:ReplacementColor, 491
Properties in VBS:RightComma, 492
Properties in VBS:Rotation, 492
Properties in VBS:RoundCornerHeight, 493
Properties in VBS:RoundCornerWidth, 494
Properties in VBS:RulerPrecisions, 496
Properties in VBS:RulerPrecisionX, 496
Properties in VBS:RulerPrecisionY, 497
Properties in VBS:SameSize, 497
Properties in VBS:ScaleColor, 498
Properties in VBS:ScaleTicks, 498
Properties in VBS:Scaling, 498
Properties in VBS:ScalingType, 499
Properties in VBS:ScalingTypeX, 499
Properties in VBS:ScalingTypeY, 500
Properties in VBS:Screen, 501
Properties in VBS:ScreenItems, 501
Properties in VBS:ScreenName, 500
Properties in VBS:Screens, 283
Properties in VBS:ScrollBars, 502
Properties in VBS:SecondNeedleHeight, 503
Properties in VBS:SecondNeedleWidth, 503
Properties in VBS:SelBGColor, 504
Properties in VBS:SelectionMode, 506
Properties in VBS:SelectionRectColor, 506
Properties in VBS:SelectionRectWidth, 507
Properties in VBS:SelectionType, 507
Properties in VBS:SelTextColor, 508
Properties in VBS:ServerData, 509
Properties in VBS:ServerNames, 509
Properties in VBS:ServerPrefix, 509
Properties in VBS:ShowBar, 510
Properties in VBS:ShowDanger, 510
Properties in VBS:ShowDecimalPoint, 510
Properties in VBS:ShowNormal, 511
Properties in VBS:ShowPeak, 511
Properties in VBS:ShowPosition, 511
Properties in VBS:ShowRulerImmediately, 512
Properties in VBS:ShowThumb, 514
Properties in VBS:ShowValuesExponentialX, 515
Properties in VBS:ShowValuesExponentialY, 515
Properties in VBS:ShowWarning, 515
Properties in VBS:SignificantMask, 516
Properties in VBS:SmallChange, 517
Properties in VBS:SourceBeginTime, 518
Properties in VBS:SourceEndTime, 519
Properties in VBS:SourceNumberOfUAValues, 519
Properties in VBS:SourceNumberOfValues, 519
Properties in VBS:SourceTagNameX, 520
Properties in VBS:SourceTagNameY, 520
Properties in VBS:SourceTagProviderDataX, 520
Properties in VBS:SourceTagProviderDataY, 521
Properties in VBS:SourceTimeRange, 521
Properties in VBS:SourceUAArchive, 521
Properties in VBS:SourceUAArchiveStartID, 522
Properties in VBS:SourceUAColumnX, 522
Properties in VBS:SourceUAColumnY, 523
Properties in VBS:SquareExtent, 523
Properties in VBS:StartAngle, 523
Properties in VBS:Statusbar, 524
Properties in VBS:StatusbarPanes, 528
Properties in VBS:StatusbarStretch, 529
Properties in VBS:Tag, 624
Properties in VBS:TagName, 531
Properties in VBS:TagPrefix, 531
Properties in VBS:TagProviderClsid, 532
Properties in VBS:Tags, 531
Properties in VBS:Template, 532
Properties in VBS:Text, 533
Properties in VBS:ThumbBackColor, 534
Properties in VBS:TicColor, 534
Properties in VBS:TicFont, 534
Properties in VBS:Ticks, 536
Properties in VBS:TicksColor, 536
Properties in VBS:TickStyle, 537
Properties in VBS:TicOffset, 535
Properties in VBS:TicTextColor, 535
Properties in VBS:TicTextOffset, 535
Properties in VBS:TicWidth, 536
Properties in VBS:TimeAxisFormat, 539
Properties in VBS:TimeAxisX, 544
Properties in VBS:TimeColumnAlignment, 545
Properties in VBS:TimeFormat, 553

Properties in VBS:TimeJump, 553
Properties in VBS:TimeJumpColor, 554
Properties in VBS:TimeOverlap, 554
Properties in VBS:TimeOverlapColor, 555
Properties in VBS:TimeRange, 555
Properties in VBS:TimeRangeBase, 556
Properties in VBS:TimeRangeFactor, 556
Properties in VBS:TimeStamp, 556
Properties in VBS:TimeZone, 558
Properties in VBS:TitleCut, 559
Properties in VBS:Titleline, 560
Properties in VBS:Toggle, 561
Properties in VBS:ToleranceHigh, 562
Properties in VBS:ToleranceLow, 562
Properties in VBS:Toolbar, 562
Properties in VBS:ToolbarAlignment, 563
Properties in VBS:ToolbarButtons, 571
Properties in VBS:ToolbarHotKeys, 572
Properties in VBS:ToolTipText, 574
Properties in VBS:Top, 575
Properties in VBS:TopConnectedConnectionPointIndex, 576
Properties in VBS:TopConnectedObjectName, 576
Properties in VBS:Transparent, 577
Properties in VBS:Trend, 577
Properties in VBS:TrendColor, 579
Properties in VBS:Type, 596
Properties in VBS:TypeAlarmHigh, 598
Properties in VBS:TypeAlarmLow, 598
Properties in VBS:TypeLimitHigh4, 599
Properties in VBS:TypeLimitHigh5, 599
Properties in VBS:TypeLimitLow4, 599
Properties in VBS:TypeLimitLow5, 600
Properties in VBS:TypeToleranceHigh, 600
Properties in VBS:TypeToleranceLow, 600
Properties in VBS:TypeWarningHigh, 600
Properties in VBS:TypeWarningLow, 601
Properties in VBS:UnitColor, 601
Properties in VBS:UnitFont, 601
Properties in VBS:UnitOffset, 602
Properties in VBS:UnitText, 602
Properties in VBS:UnselBGColor, 602
Properties in VBS:UnselTextColor, 603
Properties in VBS:UpdateCycle, 603
Properties in VBS:UpperLimit, 603
Properties in VBS:UpperLimitColor, 604
Properties in VBS:UpperLimitValue, 604
Properties in VBS:UserValue1, 607
Properties in VBS:UserValue2, 607
Properties in VBS:UserValue3, 607
Properties in VBS:UserValue4, 608
Properties in VBS:Value, 610
Properties in VBS:ValueColumnAlignment, 617
Properties in VBS:ValueMax, 623
Properties in VBS:ValueMin, 624
Properties in VBS:Visible, 624
Properties in VBS:Warning, 625
Properties in VBS:WarningColor, 625
Properties in VBS:WarningHigh, 626
Properties in VBS:WarningLow, 626
Properties in VBS:Width, 626
Properties in VBS:WindowBorder, 627
Properties in VBS:WindowsStyle, 629
Properties in VBS:WindowType, 629
Properties in VBS:WithAxes, 629
Properties in VBS:WithLabels, 630
Properties in VBS:ZeroPoint, 636
Properties in VBS:ZeroPointValue, 637
Properties in VBS:Zoom, 637
Protect VBA Code, 1562
Push Button Control, 253
putc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

Q

qsort, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
QuitHorn, 697
QuitSelected, 697
QuitVisible, 697

R

rand, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,

985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
ReadTags, 702
realloc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Reference, 1664
  Event Handling, 1664
  Methods, 1664
  Object model, 1664
  Objects and Lists, 1664
  Properties, 1664
  VBA Object Model, 1664
References, 1562
Registry2, 846
remove, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
rename, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Rename:Module, 54
Renaming, 54, 81
Renaming:Action, 81
Renaming:Procedure, 54
ReportJob, 884
rewind, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
RPTJobPreview, 847
RPTJobPrint, 848
RptShowError, 849
RulerControl, 257
Runtime, 82
Runtime:Activating global actions, 82

S

Save, 67
  Module, 52
  procedure, 52
Save As...
  Applying, 784
Save:Action, 67
Saving file as, 821
Screen Object, 140
ScreenItem, 246, 250, 257, 263, 271
  Control, 193
  Controls, 230
  Customized object, 273
  Faceplate instance, 199
  WinCC Alarm Control, 265
  WinCC AlarmControl, 236
ScreenItem Object, 134
ScreenItem:3D bar, 182
ScreenItem:Application window, 186
ScreenItem:Bar, 188
ScreenItem:Button, 214
ScreenItem:Check Box, 217
ScreenItem:Circle, 160
ScreenItem:Circle segment, 164
ScreenItem:Circular arc, 162
ScreenItem:Combobox, 202
ScreenItem:Connector, 180
ScreenItem:Double T-piece, 228
ScreenItem:Ellipse, 154
ScreenItem:Ellipse arc, 156
ScreenItem:Ellipse segment, 158
ScreenItem:Graphic Object, 200
ScreenItem:Group, 274
ScreenItem:Group Display, 208
ScreenItem:HMI Symbol Library 1.4.1, 233
ScreenItem:I/O Field, 196
ScreenItem:Line, 166
ScreenItem:ListBox, 203
ScreenItem:Multiline Text, 204
ScreenItem:Object types, 152
ScreenItem:OLE Element, 205
ScreenItem:Picture window, 191
ScreenItem:Polygon, 168
ScreenItem:Polygon Tube, 226
ScreenItem:Polyline, 170
ScreenItem:Radio Box, 219
ScreenItem:Rectangle, 172

ScreenItem:Roundbutton, 221
ScreenItem:Rounded Rectangle, 175
ScreenItem:Slider, 223
ScreenItem:Static Text, 178
ScreenItem:Status Display, 212
ScreenItem:Text list, 210
ScreenItem:T-piece, 227
ScreenItem:Tube Arc, 229
ScreenItem:WinCC Digital Analog Clock, 239
ScreenItem:WinCC Function Trend Control, 267
ScreenItem:WinCC FunctionTrendControl, 241
ScreenItem:WinCC Gauge Control, 243
ScreenItem:WinCC Online Table Control, 269
ScreenItem:WinCC OnlineTableControl, 247
ScreenItem:WinCC Push Button Control, 253
ScreenItem:WinCC Slider Control, 259
ScreenItems object (list), 138
Screens Object (List), 143
Script file, 97
Script File:Action Names, 99
Script file:open in Debugger, 100
Script File:Permissible action name length, 99
Script File:Set bookmark, 105
Script files of VBScripts, 97
SelectedStatisticArea, 712
ServerExport, 712
ServerImport, 712
Set_Focus, 1215
SetActualPointLeft, 1225
SetActualPointTop, 1226
SetAlarmHigh, 1252
SetAlarmLow, 1253
SetAlignment, 1169
SetAlignmentLeft, 1216
SetAlignmentTop, 1217
SetAssumeOnExit, 1243
SetAssumeOnFull, 1244
SetAverage, 1293
SetAxisSection, 1170
SetBackBorderWidth, 1341
SetBackColor, 1179
SetBackColor2, 1180
SetBackColor3, 1181
SetBackColorBottom, 1182
SetBackColorTop, 1183
SetBackFlashColorOff, 1203
SetBackFlashColorOn, 1204
SetBasePicTransColor, 1334
SetBasePicUseTransColor, 1335
SetBitNumber, 1245
SetBorderBackColor, 1184
SetBorderColor, 1185
SetBorderColorBottom, 1186
SetBorderColorTop, 1187
SetBorderEndStyle, 1342
SetBorderFlashColorOff, 1205
SetBorderFlashColorOn, 1206
SetBorderStyle, 1343
SetBorderWidth, 1344
SetBoxAlignment, 1345
SetBoxCount, 1227
SetBoxType, 1294
setbuf, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
SetButtonColor, 1188
SetCheckAlarmHigh, 1253
SetCheckAlarmLow, 1254
SetCheckLimitHigh4, 1255
SetCheckLimitHigh5, 1256
SetCheckLimitLow4, 1257
SetCheckLimitLow5, 1258
SetCheckToleranceHigh, 1259
SetCheckToleranceLow, 1260
SetCheckWarningHigh, 1261
SetCheckWarningLow, 1261
SetClearOnError, 1246
SetClearOnNew, 1247
SetColorAlarmHigh, 1262
SetColorAlarmLow, 1263
SetColorBottom, 1189
SetColorChangeType, 1295
SetColorLimitHigh4, 1264
SetColorLimitHigh5, 1265
SetColorLimitLow4, 1266
SetColorLimitLow5, 1267
SetColorToleranceHigh, 1268
SetColorToleranceLow, 1269
SetColorTop, 1190
SetColorWarningHigh, 1270
SetColorWarningLow, 1271
SetCursorControl, 1296
SetCursorMode, 1297
SetDirection, 1228
SetEditAtOnce, 1298
SetEndAngle, 1229
SetExponent, 1170
SetExtendedOperation, 1299
SetFillColor, 1191

SetFilling, 1201
SetFillingIndex, 1202
SetFillStyle, 1345
SetFillStyle2, 1346
SetFlashBackColor, 1207
SetFlashBorderColor, 1208
SetFlashFlashPicture, 1336
SetFlashForeColor, 1209
SetFlashPicTransColor, 1337
SetFlashPicUseTransColor, 1338
SetFlashRateBackColor, 1210
SetFlashRateBorderColor, 1211
SetFlashRateFlashPic, 1339
SetFlashRateForeColor, 1212
SetFontBold, 1218
SetFontItalic, 1219
SetFontName, 1220
SetFontSize, 1221
SetFontUnderline, 1222
SetForeColor, 1192
SetForeFlashColorOff, 1213
SetForeFlashColorOn, 1214
SetHeight, 1230
SetHiddenInput, 1248
SetHysteresis, 1300
SetHysteresisRange, 1301
SetIndex, 1340
SetItemBorderBackColor, 1193
SetItemBorderColor, 1194
SetItemBorderStyle, 1347
SetItemBorderWidth, 1348
SetLanguage, 1467
SetLeft, 1231
SetLeftComma, 1171
SetLimitHigh4, 1272
SetLimitHigh5, 1273
SetLimitLow4, 1274
SetLimitLow5, 1275
SetLimitMax, 1276
SetLimitMin, 1277
SetLink, 1292
SetLongStrokesBold, 1172
SetLongStrokesOnly, 1173
SetLongStrokesSize, 1174
SetMarker, 1277
SetMax, 1301
SetMin, 1302
SetNumberLines, 1249
SetOffsetLeft, 1303
SetOffsetTop, 1304
SetOperation, 1305
SetOperationMessage, 1306
SetOperationReport, 1306
SetOrientation, 1223
SetOutputValueChar, 1250
SetOutputValueDouble, 1250
SetPasswordLevel, 1307
SetPicDeactTransparent, 1319
SetPicDeactUseTransColor, 1320
SetPicDownTransparent, 1321
SetPicDownUseTransColor, 1322
SetPicTransColor, 1323
SetPictureDeactivated, 1324
SetPictureDown, 1325
SetPictureName, 1308
SetPictureUp, 1326
SetPicUpTransparent, 1327
SetPicUpUseTransColor, 1328
SetPicUseTransColor, 1328
SetPointCount, 1232
SetPosition, 1316
SetPressed, 1349
SetProcess, 1309
SetPropBOOL, 1330
SetPropChar, 1331
SetPropDouble
0, 1332
SetPropWord, 1333
SetRadius, 1232
SetRadiusHeight, 1233
SetRadiusWidth, 1234
SetRangeMax, 1317
SetRangeMin, 1318
SetReferenceRotationLeft, 1235
SetReferenceRotationTop, 1236
SetRightComma, 1175
SetRotationAngle, 1237
SetRoundCornerHeight, 1238
SetRoundCornerWidth, 1239
SetScaleColor, 1195
SetScaleTicks, 1176
SetScaling, 1177
SetScalingType, 1178
SetSelBGColor, 1196
SetSelTextColor, 1197
SetSmallChange, 1310
SetStartAngle, 1240
SetTagBit, 1453
SetTagBitState, 1430
SetTagBitStateWait, 1418
SetTagBitWait, 1441
SetTagByte, 1454
SetTagByteState, 1431
SetTagByteStateWait, 1419

SetTagByteWait, 1442
SetTagChar, 1455
SetTagCharState, 1432
SetTagCharStateWait, 1420
SetTagCharWait, 1443
SetTagDouble, 1455
SetTagDoubleState, 1433
SetTagDoubleStateWait, 1421
SetTagDoubleWait, 1444
SetTagDWord, 1456
SetTagDWordState, 1434
SetTagDWordStateWait, 1422
SetTagDWordWait, 1445
SetTagFloat, 1457
SetTagFloatState, 1435
SetTagFloatStateWait, 1423
SetTagFloatWait, 1446
SetTagMultiStateWait, 1424
SetTagMultiWait, 1447
SetTagPrefix, 1311
SetTagRaw, 1458
SetTagRawState, 1436
SetTagRawStateWait, 1425
SetTagRawWait, 1447
SetTagSByte, 1459
SetTagSByteState, 1437
SetTagSByteStateWait, 1426
SetTagSByteWait, 1448
SetTagSDWord, 1460
SetTagSDWordState, 1438
SetTagSDWordStateWait, 1427
SetTagSDWordWait, 1449
SetTagSWord, 1461
SetTagSWordState, 1439
SetTagSWordStateWait, 1428
SetTagSWordWait, 1450
SetTagValue, 1462
SetTagValueWait, 1451
SetTagWord, 1463
SetTagWordState, 1440
SetTagWordStateWait, 1429
SetTagWordWait, 1452
SetText, 1224
SetToggle, 1350
SetToleranceHigh, 1278
SetToleranceLow, 1279
SetTop, 1240
SetTrend, 1312
SetTrendColor, 1198
SetTypeAlarmHigh, 1280
SetTypeAlarmLow, 1281
SetTypeLimitHigh4, 1282
SetTypeLimitHigh5, 1283
SetTypeLimitLow4, 1284
SetTypeLimitLow5, 1285
SetTypeToleranceHigh, 1286
SetTypeToleranceLow, 1287
SetTypeWarningHigh, 1288
SetTypeWarningLow, 1289
SetUnselBGColor, 1199
SetUnselTextColor, 1200
setvbuf, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
SetVisible, 1313
SetWarningHigh, 1290
SetWarningLow, 1291
SetWidth, 1241
SetWindowsStyle, 1351
SetZeroPoint, 1242
SetZeroPointValue, 1314
SetZoom, 1315
ShowColumnSelection, 713
ShowComment, 713
ShowDisplayOptionsDialog, 714
ShowEmergencyQuitDialog, 714
ShowHelp, 714
ShowHideList, 715
ShowHitList, 715
ShowInfoText, 716
ShowLockDialog, 716
ShowLockList, 716
ShowLongTermArchiveList, 717
ShowMessageList, 717
ShowPercentageAxis, 718
ShowPropertyDialog, 718
ShowSelectArchive, 718
ShowSelection, 719
ShowSelectionDialog, 720
ShowSelectTimeBase, 719
ShowShortTermArchiveList, 720
ShowSort, 720
ShowSortDialog, 721
ShowTagSelection, 721
ShowTimebaseDialog, 722
ShowTimeSelection, 722
ShowTrendSelection, 723
sin, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941,

942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952,
953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963,
964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985,
986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996,
997, 998, 999, 1000, 1001, 1002, 1003
sinh, 921, 922, 923, 924, 925, 926, 927, 928, 929,
930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962,
963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,
974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,
985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
996, 997, 998, 999, 1000, 1001, 1002, 1003
Slider control, 259
Smart object, 1609
  Editing with VBA, 1610
Smart objects
  Combobox, 202
  Faceplate instance, 199
Smart Objects
  Control, 193
Smart objects:3D bar, 182
Smart objects:Application window, 186
Smart objects:Bar, 188
Smart Objects:Graphic Object, 200
Smart objects:Group Display, 208
Smart objects:I/O Field, 196
Smart objects:List Box, 203
Smart Objects:Multiline Text, 204
Smart Objects:OLE Element, 205
Smart objects:Picture window, 191
Smart objects:Status Display, 212
Smart objects:Text list, 210
Smart tag object, 145
SmartTag property, 517
sqrt, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930,
931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941,
942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952,
953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963,
964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985,
986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996,
997, 998, 999, 1000, 1001, 1002, 1003
srand, 921, 922, 923, 924, 925, 926, 927, 928, 929,
930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962,
963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,
974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,
985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
996, 997, 998, 999, 1000, 1001, 1002, 1003
Standard function
  Application, 767
  Applying, 796
  Characteristics, 767
  Deleting, 785
Standard object, 1609
  Editing with VBA, 1610
Standard Objects:Circle, 160
Standard Objects:Circle segment, 164
Standard Objects:Circular arc, 162
Standard Objects:Connector, 180
Standard Objects:Ellipse, 154
Standard Objects:Ellipse arc, 156
Standard Objects:Ellipse segment, 158
Standard Objects:Line, 166
Standard Objects:Polygon, 168
Standard Objects:Polyline, 170
Standard Objects:Rectangle, 172
Standard Objects:Rounded Rectangle, 175
Standard objects:Static Text, 178
Standard procedure, 47
Standard procedure:Use, 47
Start, 1566
  VBA editor, 1562
  VBA Macros, 1566
Starting Global Script, 30
StartStopUpdate, 723
Status bar, 775
stdio, 921, 922, 923, 924, 925, 926, 927, 928, 929,
930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962,
963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,
974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,
985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
996, 997, 998, 999, 1000, 1001, 1002, 1003
stdlib, 921, 922, 923, 924, 925, 926, 927, 928, 929,
930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962,
963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,
974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,
985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
996, 997, 998, 999, 1000, 1001, 1002, 1003
StorePicture, 1354
strcat, 921, 922, 923, 924, 925, 926, 927, 928, 929,
930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940,
941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951,
952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962,
963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,
974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,

985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strchr, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strcmp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strcpy, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strcspn, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strerror, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strftime, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
string, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strlen, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strncat, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strncmp, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strncpy, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strpbrk, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strrchr, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,

985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strspn, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strstr, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strtod, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strtok, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strtol, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
strtoul, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Symbol library, 1593
  Access with VBA, 1593
  Copying an object with VBA, 1596
  Creating a folder with VBA, 1596
  Deleting a folder with VBA, 1596
  Editing with VBA, 1596
  Paste Object into a Picture with VBA, 1599
Symbol Library, 233
Syntax, 67
Syntax:Control, 67
Syntax:Error, 67
SysFree, 920
SysMalloc, 920
system, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
System behavior, 802

T

Table Control, 269
Tag, 27, 69, 76, 2426
  Change limits with VBA, 2426
  Change type with VBA, 2426
  Creating with VBA, 2426
  Define type with VBA, 2426
  Deleting with VBA, 2426
  Read limits with VBA, 2426
  Read properties with VBA, 2426
Tag Logging, 2438
  Create archive tag with VBA, 2438
  Create process value archive with VBA, 2438
  Delete archive tag with VBA, 2438
  Delete process value archive with VBA, 2438
  Edit archive tag with VBA, 2438
  Edit process value archive with VBA, 2438
Tag Object, 146
Tag:Configure a Trigger to an Action, 76
Tag:global in VBS, 27
Tags Object (List), 149
TagScaleParam1 properties, 2306
TagScaleParam2 properties, 2306
TagScaleParam3 properties, 2306
TagScaleParam4 properties, 2306
TagSet Object (List), 151
tan, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941,

942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
tanh, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Testing, 90
Testing:with the Debugger, 90
Text library, 2465
  Changing text with VBA, 2465
  Creating language with VBA, 2465
  Creating text with VBA, 2465
  Deleting text with VBA, 2465
  Read text with VBA, 2465
  Read TextID with VBA, 2465
time, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
Timer, 69, 74
Timer:Configure a Trigger to an Action, 74
TlgGetColumnPosition, 918
TlgGetNumberOfColumns, 912
TlgGetNumberOfRows, 913
TlgGetNumberOfTrends, 913
TlgGetRowPosition, 914
TlgGetRulerArchivNameTrend, 914
TlgGetRulerTimeTrend, 915
TlgGetRulerValueTrend, 916
TlgGetRulerVariableNameTrend, 916
TlgGetTextAtPos, 917
TlgTableWindowPressEditRecordButton, 885
TlgTableWindowPressFirstButton, 886
TlgTableWindowPressInsertRecordButton, 887
TlgTableWindowPressLastButton, 888
TlgTableWindowPressNextButton, 889
TlgTableWindowPressNextItemButton, 890
TlgTableWindowPressOpenArchiveVariableSelectionDlgButton, 890
TlgTableWindowPressOpenDlgButton, 891
TlgTableWindowPressOpenItemSelectDlgButton, 892
TlgTableWindowPressOpenTimeSelectDlgButton, 893
TlgTableWindowPressPrevButton, 894
TlgTableWindowPressPrevItemButton, 894
TlgTableWindowPressRemoveRecordButton, 907
TlgTableWindowPressStartStopButton, 895
TlgTrendWindowActivateCurve, 918
TlgTrendWindowPressFirstButton, 896
TlgTrendWindowPressHelpButton, 897
TlgTrendWindowPressLastButton, 897
TlgTrendWindowPressLinealButton, 898
TlgTrendWindowPressNextButton, 899
TlgTrendWindowPressNextItemButton, 900
TlgTrendWindowPressOneToOneButton, 901
TlgTrendWindowPressOpenArchiveVariableSelectionDlgButton, 902
TlgTrendWindowPressOpenDlgButton, 902
TlgTrendWindowPressOpenItemSelectDlgButton, 903
TlgTrendWindowPressOpenTimeSelectDlgButton, 904
TlgTrendWindowPressPrevButton, 905
TlgTrendWindowPressPrevItemButton, 906
TlgTrendWindowPressPrintButton, 906
TlgTrendWindowPressStartStopButton, 908
TlgTrendWindowPressStatsResultButton, 909
TlgTrendWindowPressStatsSelectRangeButton, 909
TlgTrendWindowPressZoomInButton, 910
TlgTrendWindowPressZoomOutButton, 911
tmpfile, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
tmpnam, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
tolower, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,

974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
toolbar, 1601
Add icon, 1584
application-specific, 1573
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1582
picture-specific, 1573, 1601
Placement, 1573
Properties, 1573
user-defined, 1573
Toolbar, 1573
Working with toolbars, 781
Toolbar:GSC Diagnostics, 86
toupper, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
TraceText, 1470
TraceTime, 1470
Trigger, 69
Changing, 817
Deleting, 818
Effect on actions, 811
Picture cycle, 1653
Standard cycle, 1653
Tag, 1653
Window Cycle, 1653
Trigger type, 763, 811
Trigger:Adding "Tag" Trigger, 76
Trigger:Adding type "Timer", 74
Trigger:Deleting, 79
Trigger:editing, 78
Triggers, 74, 76, 1652
Configuring with VBA, 1652
Tube objects:Double T-piece, 228
Tube objects:Polygon tube, 226
Tube objects:T-piece, 227
Tube objects:Tube Arc, 229
TubeArcObject object, 1976
TubeDoubleTeeObject object, 1979
TubePolyline object, 1981
TubeTeeObject object, 1984

U

ungetc, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
UnhideAlarm, 725
UnlockAlarm, 725
User interface language, 1570
Access with VBA, 1570
UserArchiveControl, 263
User-defined menu, 1566, 1573
Add menu entry, 1577
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1575
creating multiple languages, 1580
Placement, 1573
Properties, 1573
User-defined toolbar, 1566, 1573
Add icon, 1584
Assigning help text, 1586
Assigning status text, 1586
Assigning VBA macro, 1589
Configuring, 1572
Creating, 1582
Placement, 1573
Properties, 1573

V

Validity range, 772
VB action, 1650
Configuring on an event with VBA, 1650
VB Scripts, 1642
Dynamize property with VBA, 1642

VBA, 1560
Access to a copy of the pictures, 1601
Access to external applications, 1658
Access to group objects, 1618
Access to layers, 1601
Access to objects in the Graphics Designer, 1606
Access to other programs, 1658
Access to the component library, 1593
Access to the symbol library, 1593
Accessing MS Excel with VBA, 1659
Action configuring, 1644
Add menu entry to user-defined menu, 1577
Adding a New Icon to the used-defined Toolbar, 1584
Alarm Logging, 2478
Assign help text to user-defined menu, 1586
Assign help text to user-defined toolbar, 1586
Assigning VBA macro to user-defined menu, 1589
Assigning VBA macro to user-defined toolbar, 1589
C Scripts (Delimitation), 1561
Configure user-defined menu, 1572
Configure user-defined toolbar, 1572
Configuring a Direct Connection, 1645
Configuring C Action on an Event, 1648
Configuring Dynamics in a Property with C Script, 1639
Configuring Dynamics in a Property with Dynamic Dialog, 1636
Configuring Dynamics in a Property with VB Script, 1642
Configuring event-controlled actions, 1644
Configuring Multiple Languages, 1570
Configuring triggers, 1652
Configuring VB Action on an Event, 1650
Controlling the visibility of layers, 1603
Create an application-specific menu, 1575
Create application-specific toolbar, 1582
Create user-defined menu, 1575
Create user-defined menu in multiple languages, 1580
Create user-defined toolbar, 1582
Creating a group object, 1620
Customized Object, 1626
Deleting Group Objects, 1620
Dynamic Wizards (Delimitation), 1561
Dynamization, 1631
Edit component library, 1596
Edit layers, 1603
Edit Smart object, 1610
Edit standard object, 1610
Edit Windows object, 1610
Editing a copy of a picture, 1605
Editing a group object, 1620
Editing Customized Objects, 1628
Editing objects in a group object, 1623
Editing objects in Graphic Designer, 1606
Editing objects in Graphics Designer, 1606
Editing picture, 1601
Editing symbol library, 1596
Picture, 1599
Paste Object from symbol library into a Picture, 1599
Paste Object into Picture, 1609
picture-specific VBA code, 1562
Project language, 1570
project-specific VBA Code, 1562
Tag Logging, 2438
Tag Management, 2426
Text library, 2465
Ungrouping Group Object, 1620
Usage, 1561
User interface language, 1570
User-defined menu, 1573
User-defined toolbar, 1573
VB Scripts (Delimitation), 1561
VBA Code, 1565
Execution sequence, 1562
Exporting, 1565
global, 1562
Importing, 1565
Organization within the WinCC Project, 1562
Password protection, 1562
picture-specific, 1562
project-specific, 1562
protecting, 1562
References, 1562
VBA editor, 1562
Start, 1562
VBA events
see VBAEvents:VBA Event Handling, 1664
VBA in the Graphics Designer, 1567
VBA Macros, 1566
Specifics during the execution, 1562
Version, 1566
VBA Object Model, 1664
VBA Reference, 1664
Event Handling, 1664
Events, 1664
Methods, 1664
Object model, 1664
Properties, 1664
VBA Object Model, 1664

VBS, 27, 734
Application Scenarios, 14
Basics, 760
Examples, 755
Examples in WinCC, 734
Methods, 638
Object model, 109
Objects, 113
Reference, 109
Target Group of the Documentation, 14
VBS:Action, 21
VBS:CrossReference, 25
VBS:Editors, 28
VBS:Module, 18
VBS:Object Types, 152
VBS:Procedure, 18
VBS:Properties, 276
VBS:Standard Functions, 45
VBS:Using Global Tags, 27
VBScript Files, 97
VBScript Files:Structure, 97
VBScript:activate in Runtime, 82
VBScript:debugging, 90
VBScript:open in Debugger, 100
VBScript:print, 108
vfprintf, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003
View, 783
Setting different views, 783
Visual Basic Script in WinCC, 14
Visual Basic Scripts, 14
vsprintf, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003

W

WinCC, 14, 152
Script Languages, 14
Visual Basic Scripts, 14
WinCC Alarm Control, 265
WinCC AlarmControl, 236
WinCC MediaControl, 246
WinCC coding rules, 805
WinCC:Graphic object types, 152
WinCC:WinCC Digital Analog Clock, 239
WinCC:WinCC Function Trend Control, 267
WinCC:WinCC FunctionTrendControl, 241
WinCC:WinCC Gauge Control, 243
WinCC:WinCC Online Table Control Online, 269
WinCC:WinCC Online Trend Control, 271
WinCC:WinCC OnlineTableControl, 247
WinCC:WinCC OnlineTrendControl, 250
WinCC:WinCC Push Button Control Controls:WinCC Push Button Control, 253
WinCC:WinCC RulerControl, 257
WinCC:WinCC UserArchiveControl, 263
Window, 84, 87
Windows object, 1609
Editing with VBA, 1610
Windows Objects:Button, 214
Windows Objects:Check Box, 217
Windows Objects:Radio Box, 219
Windows Objects:Roundbutton, 221
Windows Objects:Slider, 223
WriteTag, 729
Writing tag values, 739

Z

ZoomArea, 729
ZoomInOut, 730
ZoomInOutTime, 730
ZoomInOutValues, 730
ZoomInOutX, 731
ZoomInOutY, 731
ZoomMove, 732