Scripting for WinCC and WinCC Professional

WinCC V7, WinCC Professional

https://support.industry.siemens.com/cs/ww/en/view/109773206

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG (“Siemens”). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens’ products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens’ products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer’s exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of Contents

1 Introduction

Required knowledge

Fundamental knowledge of WinCC V7 or WinCC Professional is required.

Notes

The fundamentals for the products used in this application example are explained in the following SITRAIN courses:

- WinCC V7 Fundamentals course/System course
  Article ID: 109758633
- WinCC Professional Fundamentals course/System course
  Article ID: 109758618

1.1 Overview

The application example should provide the necessary fundamentals and knowledge for the advanced dynamization of screens and the processing of background tasks in WinCC V7 and WinCC Professional in the programming languages VBS and ANSI-C.

The application example does not provide the knowledge for programming in the respective scripting languages.

The application example addresses the following key topics:

- Useful information about scripting in WinCC
- VBS for dynamization (RT V7, RT Professional)
- ANSI-C for dynamization (RT V7, RT Professional)
- VBA (WinCC V7 only)

Notes

Details about VBA (WinCC V7 only) and the Runtime API are not discussed in detail. This document provides only a brief overview of their possible applications.

1.2 Principle of Operation

You can use scripts in process operation (VBS and ANSI-C) in the following places in WinCC:

- In the "Graphics Designer", you can configure screen-dependent actions:
  - Property scripts that return dynamic properties of graphic objects (such as object color).
  - Event scripts that are called when events occur in Runtime (for example, a mouse click).
- In the "Global Script Editor"
  Here, you configure screen-independent actions (VBS and ANSI-C) and procedures (VBS) or functions (ANSI-C).
- In user-defined menus and toolbars (VBS only)
  Here you configure procedures that are accessed via the menu and toolbars in Runtime.

During project creation, VBA scripts help you automate the creation of process screens in the Graphics Designer.

1.3 Components Used

The following hardware and software components were used to create this application example:

Table 1-1

Component	Quantity	Article number	Notes
Engineering Station	1		Development computer
SCADA WinCC V7	1	6AV63.1-....7-....	Alternative to SCADA WinCC Professional
SCADA WinCC Professional	1	6AV2103-0….-0...	Alternative to SCADA WinCC V7

This application example consists of the following components:

Table 1-2

Component	File name	Notes
Documentation	109773206_Scripting_WinCC_en.pdf	This document

2 Useful Information

The programming languages VBS and ANSI-C offer the possibility of advanced dynamization of screens and the processing of background tasks in WinCC V7 and WinCC Professional.

However, you should avoid using scripts if possible, and instead use

- animations,
- direct connections, or
- the Dynamic dialog,

as these offer better Runtime performance.

The FAQ "How to close a faceplate in WinCC Runtime using an additional button" shows an example of replacing a C script or VB script with a higher-performance direct connection:

https://support.industry.siemens.com/cs/ww/en/view/109769328

Notes

You can find additional information on animations and direct connections in the manual "WinCC V7.5 SP1: Working with WinCC" –

"How to Animate an Object":

https://support.industry.siemens.com/cs/ww/en/view/109773058/126899021579

"How to Configure a Direct Connection":

https://support.industry.siemens.com/cs/ww/en/view/109773058/103315496587

2.1 Fundamentals

2.1.1 VBS

VBS (Visual Basic Script) is a scripting language developed by Microsoft. It is closely related to VB (Visual Basic) and VBA (Visual Basic for Applications).

VBS is used in WinCC to dynamize the graphical objects in process operation (Runtime):

- Tags
  Tag values can be read and written, e.g. to specify tag values for the controller via a mouse click event on a button.
- Objects
  Object properties can be made dynamic using actions, and actions can be triggered by events influencing objects.
- Screen-independent actions
  Screen-independent actions can be triggered cyclically or according to tag values, e.g. for the daily transfer of values into an Excel table.

2.1.2 ANSI-C

The programming language C was developed in 1972 and published in 1989 by the association ANSI (American National Standard Institute) under a single standard as ANSI-C.

ANSI-C is used in WinCC to dynamize the graphical objects in process operation (Runtime):

- Tags
  Tag values can be read and written, e.g. to specify tag values for the controller via a mouse click event on a button.
- Objects
  Object properties can be made dynamic using actions, and actions can be triggered by events influencing objects.
- Screen-independent actions
  Screen-independent actions can be triggered cyclically or according to tag values, e.g. for the daily transfer of values into an Excel table.

2.1.3 VBA

VBA (Visual Basic for Applications) is used to create process screens in the Graphics Designer. This allows you to simplify and automate the project configuration.

This includes:

- Adjustment of the Graphics Designer
- Editing screens
- Editing objects
- Dynamizing with VBA
- Creating new tags, archive tags, and messages
- Access to external applications

Notes VBA is not supported in WinCC Professional.

2.2 Delimitation

2.2.1 VBS and ANSI-C

You can use VB scripts in WinCC parallel to C scripts, but you cannot mix the script types:

- VB scripts and C scripts can be configured within a screen and project.
- C scripts cannot be accessed within VB scripts and vice versa.
- In VBS, only internal interfaces to tags and screen objects are available, whereas in the C environment, you can also access other WinCC subsystems (such as the report system).

2.2.2 VBS and VBA

VBA is used in WinCC during the configuration in order to adapt the Graphics Designer to your individual requirements and to simplify and automate configuration. VBA programs only run in the WinCC configuration environment.

In contrast to VBA, VB scripts only run in WinCC Runtime and, from there, enable access to graphic objects and tags. Objects and screens 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 include, for example:

- VBS was developed for use on the internet, whereas VBA was developed for the automation of software applications.
- The data type of VBS tags is always VARIANT. VBA, on the other hand, differentiates between the individual data types such as INT, DOUBLE, STRING, etc.
- Certain language constructs from VBA have been removed from or added to VBS.
- The error handling in VBS is handled differently than in VBA, see section 4.3.

2.2.3 Comparing VBS to ANSI-C

In principle, no general recommendation can be made regarding preference between VBS and ANSI-C. Instead, which language should be used depends on the individual case.

Performance

The following measured values show the performance differences in milliseconds between VB scripting and C scripting based on typical tasks. The values depend on the hardware used.

Table 2-1

Purpose	VBS	ANSI-C
Set colors from 1,000 rectangles.	220	1,900
Set output values from 200 I/O fields.	60	170
Select a screen with 1,000 static texts which determine the object name and are issued as the return value.	460	260
Read 1,000 internal tags.	920	500
Re-read 1,000 internal tags.	30	120
Perform 100,000 calculations.	280	70

Scripts for 100,000 calculations in the example:

```
' VBS
For i=1 To 100000
value=Cos(50)*i
Next
```

```
// ANSI-C
for(i=1;i<=100000;i++)
{
dValue=cos(50)*i;
}
```

Limitations for WinCC/WebUX

WinCC/WebUX provides a solution for operator control and monitoring of the automation system independent of device and browser. The employed Web technology results in restrictions to the WinCC basic system:

- ANSI-C is not supported.
- VBS is supported with restrictions.
- Dynamic dialog:
  - Dynamizing via the Dynamic dialog are converted into VBS.
  - Complex formulas and scripts are not supported.
- Scripts are not executed locally on a WebUX client. Scripts that run locally on a WinCC client are also executed on the WebUX server when accessed via WebUX.

Limitations for faceplate types

The following dynamization options are possible for a faceplate type:

- Tag connection to tags:
  - Interface tags
  - Structure type elements
  - Faceplate tags
- Animation
- VB scripts
  - The VB scripts are available only in the faceplate type.
  - You cannot use VB scripts to access data outside of the faceplate type, only the interface can be accessed.
- ANSI-C is not supported.

2.2.4 VBA in Comparison

- VBS and ANSI-C scripts are only active in Runtime and are used to dynamize screen and object properties as well as in action configuration.
- The Dynamic Wizards are not completely replaceable by VBA. However, VBA offers the possibility to create new functionalities.
- In contrast to ODK, VBA is characterized by simple object-oriented access to the objects of the Graphics Designer. ODK are function commands that provide access to all WinCC functionalities, both in the Configuration System and in Runtime.

3 Functionality Details

3.1 Password Protection

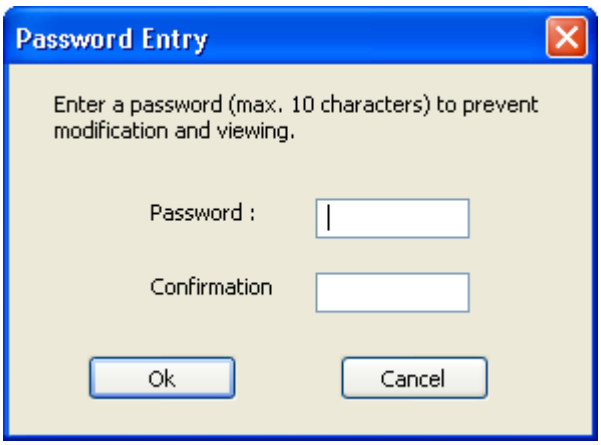
3.1.1 Password Protection (WinCC V7)

VBS and ANSI-C

You can protect your global scripts (functions and actions) against changes and viewing with a password. The password is part of the accompanying information.

1. Open Global Script.
2. Open the script that you want to password-protect in the Navigation window.
3. Click the "Info/Trigger" button in the "Edit" toolbar or select the "Info" context menu command. The "Properties" dialog is opened.
4. Check the "Password" check box.
5. Confirm by clicking "Change".

Figure 3-1



6. Enter the password in the "Password" field.
7. Enter the password again in the "Confirmation" field.
8. Confirm your entries with "OK".
9. Click "OK" to close the dialog.

Notes

A password-protected script can only be opened in the editing window by entering the password.

If you want to remove the password protection, uncheck the "Password" checkbox after you have opened the script.

VBA

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.

3.1.2 Password Protection (WinCC Professional)

To configure knowledge protection for user-defined functions, proceed as follows:

1. Select the user-defined function without knowledge protection that you want to protect.
2. Select "Knowledge protection" from the "Edit" menu.
   The "Knowledge protection" dialog opens.
3. Click "Define".
   The "Define password" dialog is opened.
4. Enter a password in the "New" field.
5. Repeat the password in the "Confirm" field.
6. Confirm the entry with "OK".
7. Close the "Knowledge protection" dialog by clicking "OK".

3.2 Storage Location (WinCC V7)

3.2.1 VBS Storage Location

Standard modules

Standard modules contain procedures that are available across projects.

Standard modules are stored in the WinCC file system under:
<WinCC installation directory>\ApLib\ScriptLibStd\<module name>.bmo

CAUTION	**If you reinstall WinCC, the standard modules in the WinCC installation directory are deleted during the installation.** **Save the module files in another directory before the new installation and copy them back into the WinCC directory after the installation.**

Project modules

Project modules contain project-specific procedures.

Project modules are stored in the WinCC file system under:
<Project directory>\ScriptLib\<module name>.bmo

Notes	Since the project modules are stored in the project directory, they are copied when the WinCC project is copied.

Actions

Actions of the Global Script are stored in the WinCC file system under:

<Project directory>\ScriptAct\<action name>.bac

Notes	Since the actions are stored in the project directory, they are also copied when copying a WinCC project. Each action is saved in a separate file.

3.2.2 ANSI-C Storage Location

Project functions

Project functions are created by the user and are project-specific.

Project functions are stored in the WinCC file system under:
<Project directory>\library\<function name>.fct

Notes	Since the project functions are stored in the project directory, they are also copied when copying a WinCC project.

Standard functions

Standard functions are provided by WinCC and may be modified by the user or supplemented with new ones. They are stored in the WinCC file system under:
<WinCC installation directory>\ApLib\<subdirectory>\<function name>.fct

Depending on the function, the following subdirectories are provided:

- Alarm
- Graphics
- Report
- TagLog
- WinCC
- Windows

CAUTION	**If you reinstall WinCC, the standard functions in the WinCC installation directory are deleted or overwritten during installation.** Save modified or new function files in a different directory before reinstalling and copy them back into the WinCC directory after the installation.

Internal functions

Internal functions are exclusively provided by WinCC and cannot be changed or supplemented by the user.

They are stored in the WinCC file system under:
<WinCC installation directory>\ApLib\<subdirectory>\<function name>.icf

Depending on the function, the following subdirectories are provided:

- Allocate
- C-bib
- Graphics
- Tag
- WinCC

Local actions

Local actions are created by the user and are computer-specific.

Local actions are stored in the WinCC file system under:
<project directory>\<computer name>\Pas\<action name>.pas

Notes Because the local actions are stored in the project directory, they are also copied when copying a WinCC project.

Global actions

Global actions are created by the user and are executed in a Client–Server project on all computers.

Global actions are stored in the WinCC file system under:
<Project directory>\Pas\<action name>.pas

Notes Because the global actions are stored in the project directory, they are also copied when copying a WinCC project.

3.2.3 VBS Storage Location

Global VBA code

Global VBA code refers to VBA code that you write in the VBA Editor in the document "GlobalTemplateDocument" and that should be available in all WinCC projects on your computer.

Global VBA code is stored in the WinCC file system under:
<WinCC installation directory>\Templates\@GLOBAL.PDT>

Notes

- If you need the global VBA code on a different computer, use the export and import functions in the VBA Editor.
- When you perform an update installation, your global "@GLOBAL.PDT" template is saved in the "@GLOBAL.SAV" backup file. Your VBA code from the old global template is not automatically applied to the new 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 in the new global template, open the VBA Editor in the Graphics Designer and copy the VBA code.
2. Close WinCC.
3. Open the directory …\Siemens\WinCC\Templates in the Windows Explorer.
4. Delete the new global template "@GLOBAL.PDT".
5. Rename the backup file "@GLOBAL.SAV" to "@GLOBAL.PDT".
6. If you have copied VBA code from the new global template, open the VBA Editor in the Graphics Designer and paste the copied VBA code.

Project-specific VBA Code

Project-specific VBA code refers to VBA code that you write in the VBA Editor in the "ProjectTemplateDocument". Insert the VBA code that you want to use in all screens of the opened project into the "ProjectTemplateDocument".

Project-specific VBA code is stored in the WinCC file system under:
<Project directory>\Templates\@PROJECT.PDT>

Notes

- If you need the project-specific VBA code on another computer, use the export and import functions of the VBA Editor.
- The "@PROJECT.PDT" file has a reference to the "@GLOBAL.PDT" file. Therefore, you can access functions and procedures that you have saved in the file "@GLOBAL.PDT" directly in the "ProjectTemplateDocument".

Screen-specific VBA code

Screen-specific VBA code refers to VBA code that you write in the VBA Editor in the document "This Document" of the respective screen.

This VBA code is saved as a PDL file together with the screen.

Notes

The PDL file has a reference to the "@PROJECT.PDT" file.

Therefore, you can access functions and procedures stored in the file "@PROJECT.PDT" directly from the PDL file.

However, you do not have access to functions or procedures that are stored in the "@GLOBAL.PDT" file.

3.3 Cross-References (CrossReference Editor)

Tag trigger

Tag triggers from actions in the Graphics Designer can be "linked" with CrossReference, i.e. replaced by other tags at all or selected points.

Actions

- All the actions used in a screen can be displayed via the screen properties:
  - Select the screen in the WinCC Explorer
  - Select the "Properties" command in the context menu
  - After double clicking an entry, detailed information on the type of the dynamization appears.
- It is also possible to display all the tags and screens used in actions by means of the WinCC CrossReference.
- With CrossReference, you can also link the tag bindings of the Graphics Designer actions.

Notes

In order to be able to display or link the tags and screens used in actions via WinCC CrossReference, they must be explicitly declared. The corresponding coding rules are explained below.

Tags (VBS)

All tags addressed with the following standard formulation are automatically compiled by WinCC CrossReference and then listed in the screen properties:

```
HMIRuntime.Tags("Tagname")
```

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

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

Screens (VBS)

All screens addressed with the following standard formulation are automatically compiled by WinCC CrossReference and then listed in the screen properties:

```
HMIRuntime.BaseScreenName = "Screenname"
```

If screens are addressed with different formulations in the code, this can be indicated by the following CrossReference section:

```
' VBS
' WINCC:SCREENNAME_SECTION_START
Const ScreenNameInAction = "ScreenName"
' WINCC:SCREENNAME_SECTION_END
```

Tags and screens (ANSI-C)

The action's code begins with two sections:

- In the first section, all tags used must be declared.
- In the second section, all used screen names must be declared.

Notes

- Both sections are already predefined in the form of comments when creating a new action.
- You cannot enter any additional commands within the sections.

The declaration of tags and screens is done by adding appropriate #define commands to the given sections:

```
// ANSI-C
// 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
```

The functions for reading and writing the tags and using the screen names must then be accessed with the defined names:

```
// ANSI-C
GetTagDWord(ApcTagName1);
OpenPicture(ApcPicName1);
SetPictureName(ApcPicName2,"PictureWindow1",ApcPicName3);
```

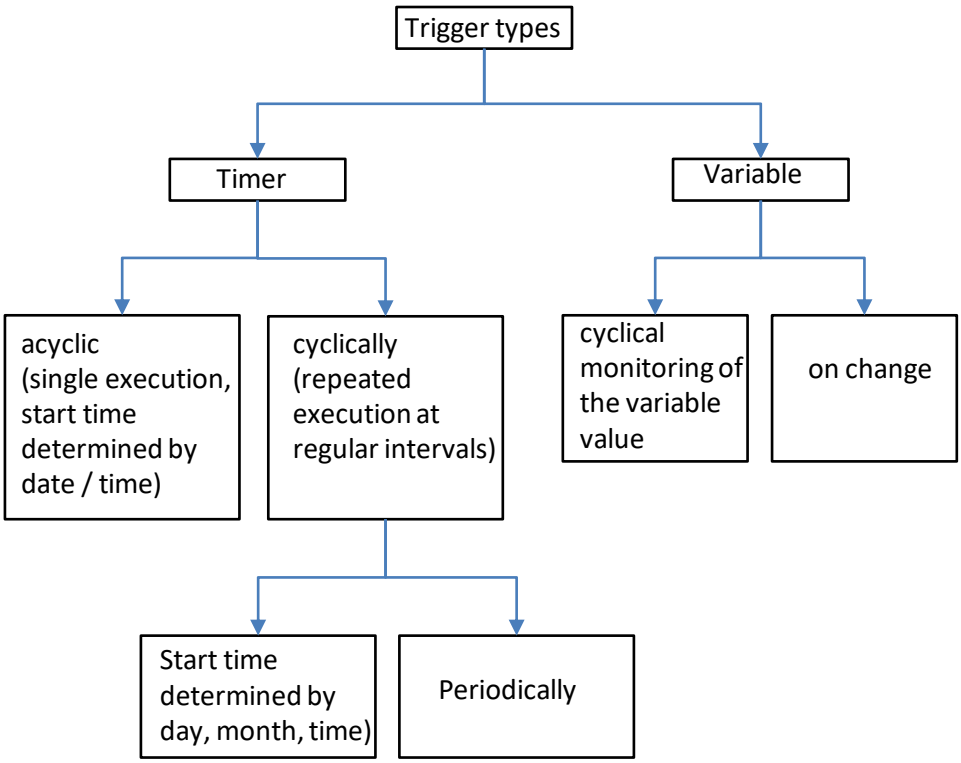
3.4 Triggers

A trigger forms the triggering event for accessing the action. Actions without a trigger are not executed.

The following trigger types are available in WinCC:

- Timers
  Acyclic or cyclical triggers, e.g. either when accessing a screen or every hour.
- Tag
  Value changes (cyclical monitoring or "upon change" for internal tags)
- Events
  Changes to object properties (e.g. color change) or events on an object (e.g. mouse click).

Figure 3-2



Acyclic triggers

These contain a date and time specification. The action specified by such a trigger is executed once at the date and time specified.

Cyclical triggers

These consist of the specification of a time interval and its start. Time-controlled triggers are used for actions in Global Script and for actions to dynamize graphic objects.

Tag trigger

These consist of one or more specified tags. The action associated with such a trigger is executed 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:

- Cyclical query of the tag value
  Specify a standard cycle. At the selected intervals (e.g. every 2 seconds), the value of the tags is queried. The action is triggered when the system detects a change in the tag value.
- 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 in Global Script and for actions to dynamize graphic objects.

Event trigger

When you configure an action to an event on a graphic object, the action is event-triggered, e.g. when you click with the mouse or change the background color by means of another action.

Notes

As of WinCC V7.0, the trigger type "Animation cycle" is available for animations 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.

3.5 Timing

Actions from the Global Script and actions in the Graphics Designer are processed independently in Runtime. There is no common data area between the two Runtime systems.

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

In Graphics Runtime

- If the processing of cyclical actions in screens is hindered, for example, by a heavy system load or other actions, the action is executed once again at the next possible opportunity. Unexecuted cycles are not held in a queue, but are instead discarded.
- VB scripts still running after a screen change are automatically terminated 1 minute after the screen change.
- VB scripts that are still running when you exit Runtime will be terminated after 5 seconds.

In Global Script

- Screen-independent actions from Global Script are executed sequentially in Runtime when triggered.
- If an action is triggered while another action is in progress, the second action is held in a queue until execution is possible.

Queues

WinCC has several queues (queues for intermediate storage of scripts), which are processed independently of each other:

- In the Graphics Runtime, there are two queues each for VBS and ANSI-C:
  - A queue for editing scripts of properties (e.g. color change when the value of a tag changes).
  - A queue for processing scripts via events (e.g. mouse click).
- In global scripting, there are two queues for ANSI-C:
  - One queue for cyclical triggers (periodically, e.g. "1 second").
  - One queue for tag triggers (e.g. when the value of a tag changes).
- In global scripting, there is a common queue for VBS for both trigger types.

Notes You get the best performance by distributing the script traffic over as many queues as possible. Therefore using a mix of VBS and ANSI-C is advantageous.

3.6 Tips – Performance

Behavior in actions with a tag trigger

The tag triggers should preferably be used instead of cyclical triggers:

- With cyclical actions, the action is always executed and thus places a load on the system.
- The tag trigger, on the other hand, only executes the action if a change in value is detected during cyclical scanning of the tags.
- All tags contained in the tag trigger are already recognized before the script is called and are collectively registered with the communication channel together with the specified monitoring time. The respective value is then directly transferred within the script when accessed, which results in a significant performance gain.

Triggering on screen transition

For dynamization with access to screen objects, note that a script that is being processed is not automatically terminated by deselecting the screen. This can cause access to objects addressed in the script to fail.

Modal dialog

Please note that a modal dialog, e.g. the Windows "Message Box" dialog box, stops the processing in Runtime.

An alternative to the Windows dialog box are the faceplates that you may configure directly in WinCC. You can find the relevant post here:

https://support.industry.siemens.com/cs/ww/en/view/268859

Time-out

If another application is accessed in the Global Script, there may be a delay in starting or stopping this application. If this results in longer waiting times, a time-out is used to prevent the Global Script Engine from being blocked.

Modules (VBS)

- The more modules must be loaded when accessing a screen, the worse the performance in Runtime.
- The larger a module is, i.e. the more procedures it contains, the longer the loading time for the module.
- Organize the modules sensibly, e.g. a module with procedures for a specific system part/screen.

TagSet (VBS)

The object "TagSet" enables simultaneous access to several tags via one call. This is done with better performance and less communication load than with single access to various tags.

The following example shows how to create a "TagSet" object, add tags, and write values:

```
' VBS
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
```

Direct read (VBS)

Please note that this call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, amongst other things.

Avoid calling ".Read(1)" in cyclical VBS actions, as this can lead to performance problems.

Direct read (ANSI-C)

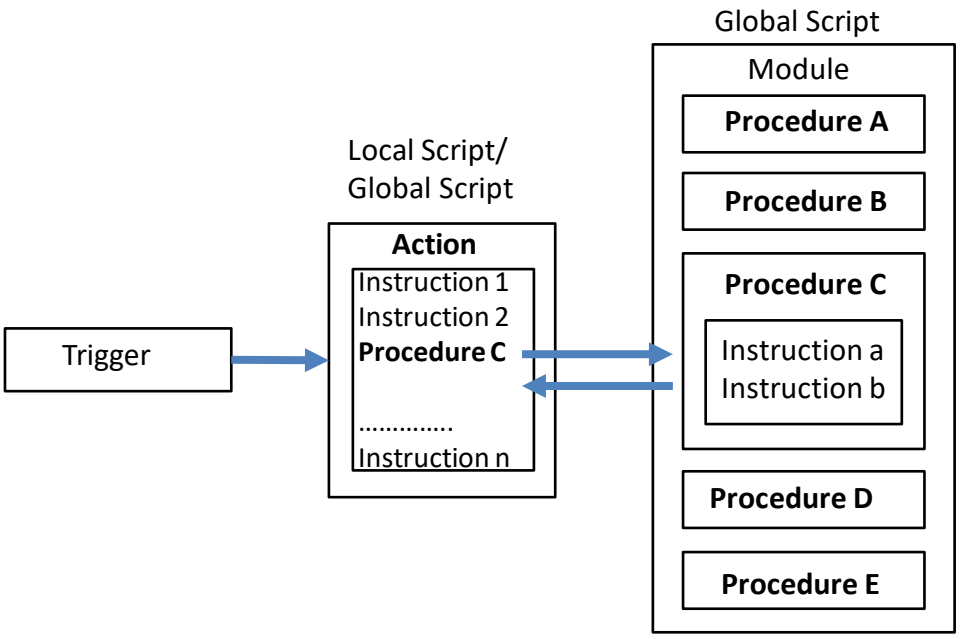
Please note that this call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, amongst other things.

Avoid calling "GetTagWait" in cyclical C actions, as this can lead to performance problems.

4 VBS

4.1 Interface Description

Figure 4-1



4.1.1 Actions

- Actions are always activated by a trigger – namely a triggering event.
- You configure actions as properties of graphic objects, events that occur for a graphic object, or globally in the project.
- Actions are used to call code that is used several times in the form of procedures.
- Actions in the Global Script can be protected against modification and viewing by means of a password.

Notes

Global actions are only valid in the project in which they were defined.

- **For clients in a multi-user system**
  All global actions configured on a server are also executed on a client.
- **For clients in a distributed system**
  If you want to use actions on a client computer, copy the action files into the corresponding project directory on the client.

4.1.2 Procedures

Properties

- A procedure corresponds to a function in C.
- Procedures are configured globally in the project.
- Procedures are used to store code that you want to use at several places in your project configuration.
- Procedures are stored in a module.
- Access code within an action or another procedure by calling the procedure name.
- Procedures can be created in WinCC with or without return values.
- Procedures do not have their own trigger; they are always called via an action.
- Procedures can be protected against modification and viewing by means of a password.

Standard procedures

Standard procedures apply globally to projects located on the computer on which they were created.

Project procedures

Project procedures can only be used within the project in which they were created.

4.1.3 Modules

Properties

- You configure modules globally in the project.
- It is advantageous to compile related procedures to units in modules.
- For example, you can create modules for procedures that are used in a certain screen or belong to a certain topic.
- Modules have the file extension *.bmo.
- Modules can be protected against modification and viewing by means of a password.

Notes

- The more modules must be loaded when accessing a screen, the worse the performance in Runtime.
- The larger a module is, i.e. the more procedures it contains, the longer the loading time for the module.

Standard modules

Standard modules contain procedures that are available across projects.

Project modules

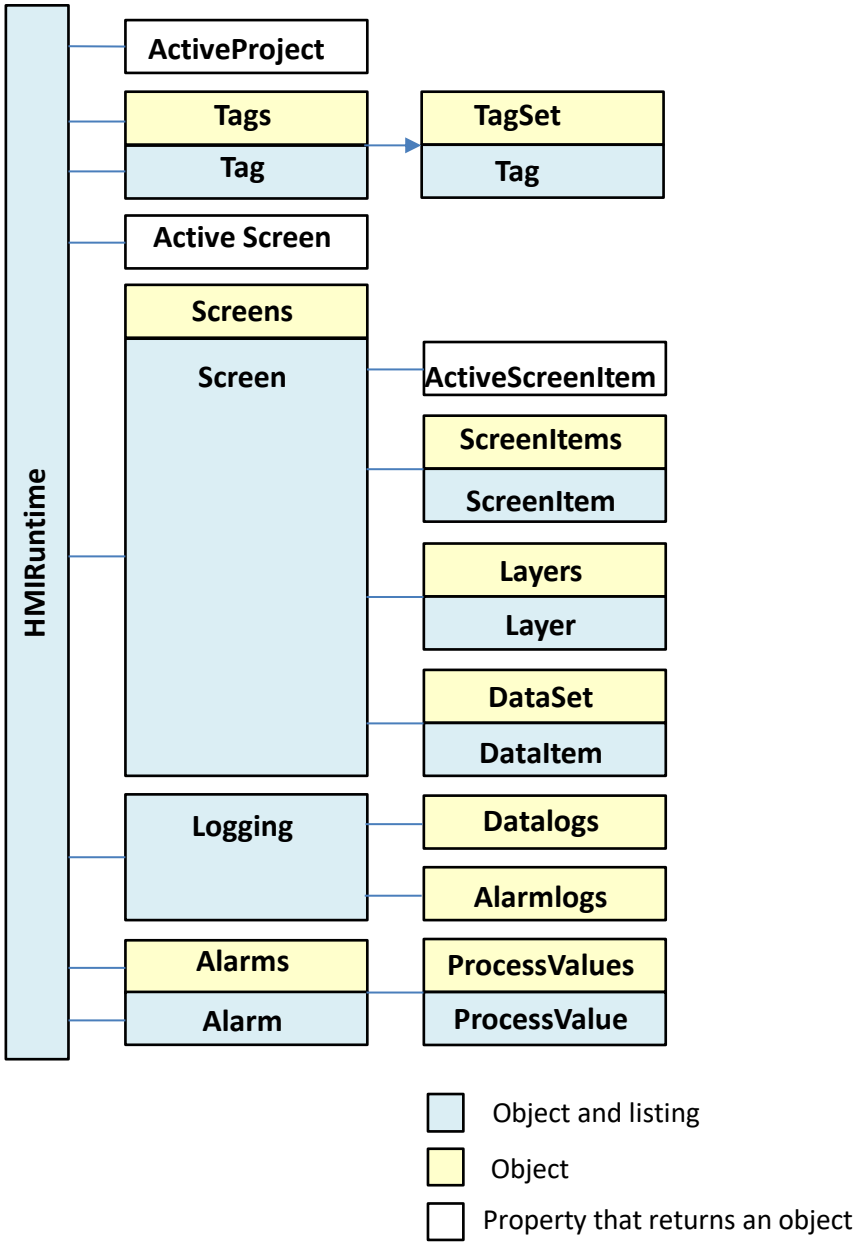
Project modules contain project-specific procedures.

4.2 Integration in the User Project

4.2.1 Object Model

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

Figure 4-2



Notes

The VBS object model is not valid for WinCC for a faceplate type.

The VBS object model of the faceplate type allows you to access only the graphic objects, tags, and interface 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, screens, layers, and tags.

Properties

Using the properties of the individual objects, you can selectively change graphic objects and tags in Runtime, e.g. activate a control element with a mouse click or trigger a color change when a tag value is changed.

Methods

With the methods that you apply to the individual objects, you can, for example, read out tag values for additional processing or output diagnostic messages in Runtime.

Navigation in object models

Access is made to objects in the VBS object model in hierarchical sequence. For example, access to a screen element within a screen is done via its parent object (the surrounding screen).

Access to graphic objects

In WinCC, access to the screens, layers, and graphic objects in Runtime is via the superordinate object "HMIRuntime". Access to objects and layers is always made via the 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 reset.

Notes

The listings of the WinCC object model behave like the standard collections of VBS, with the exception of the "Tags" listing, which has no Enum functionality.

Available objects

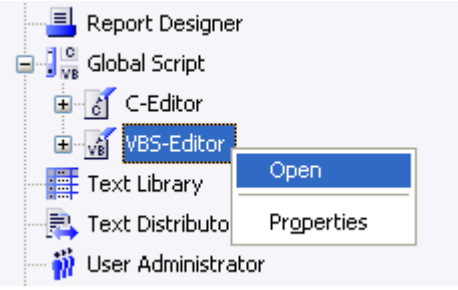
- Alarm
- Alarms
- AlarmLogs
- DataItem
- DataLogs
- DataSet
- HMIRuntime
- Item
- Layer
- Layers
- Logging
- ProcessValues
- ProcessValue
- Project
- ScreenItem
- ScreenItems
- Screen
- Screens
- Tag
- Tags
- TagSet

4.2.2 Editor

You can program VB scripts in two places in WinCC:

- The Global Script Editor is used to program actions and procedures that you can access throughout the project and that can run independently of the screen currently displayed.

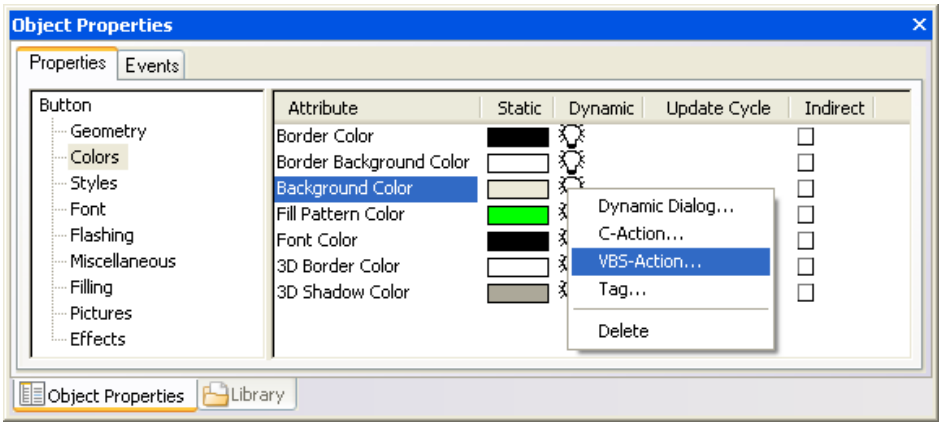
  Figure 4-3



Notes The Global Script Editor is started via the context menu command "Open" in the project window of WinCC Explorer.

- The Graphics Designer allows you to program actions on object properties or events on graphic objects.

  Figure 4-4



Notes You can access the action editor in the Graphics Designer via the context menu in the properties dialog box of a graphic object.

4.2.3 Creating and Editing Procedures

You can configure procedures in Global Script:

- 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 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.

Use

- Procedures are used for the central creation and maintenance of codes which are to be implemented at several points throughout the configuration.
- You can write and save the code in a procedure and then call this procedure with the current parameters in either actions (both screens and global actions) or in other procedures instead of repeatedly entering the same code.

Notes

You can find additional information on the relationships between actions, procedures, and modules in section 4.1, "Interface Description".

4.2.4 Creating and Editing Actions

Actions can be configured in Global Script and in the Graphics Designer:

- Use Global Script to configure global actions which can be executed independently of the screen currently open.
- Graphics Designer is used to configure actions related to graphic objects which should be executed when the screen is opened in Runtime or when the configured trigger occurs.

Use in Global Script

You can use screen-independent actions in Global Script as cyclical actions:

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

Use in the Graphics Designer

You can use actions as dynamization of an object property in the Graphics Designer:

```
' VBS
Function BackColor_Trigger(ByVal Item)
BackColor_Trigger = RGB(125,0,0)
End Function
```

You can use actions triggered by an event on an object in the Graphics Designer:

```
' VBS
Sub OnClick(ByVal Item)
Item.BackColor = RGB(255,0,0)
End Sub
```

Notes You can find additional information on the relationships between actions, procedures, and modules in section 4.1, "Interface Description".

4.2.5 Reading Tags

Basically, a distinction is made between two types of tag updating:

Read from the tag screen (asynchronous call)

- When reading from the tag screen, the tag is registered and, from then on, cyclically requested from the automation system.
- The login cycle is dependent on the configured trigger.
- The value is read from the tag screen by WinCC.
- When a screen is deselected, the displayed tags are deregistered again.
- The call is faster compared to direct reading. However, the first call takes longer, since the value must first be read from the AS and registered.
- The duration of the call does not depend on the bus load or on the AS.

```
' VBS
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read
```

Direct reading from the AS (synchronous call)

- The tag is not registered cyclically, the value is requested from the AS one time only.
- The current value is read explicitly from the AS.
- The call takes longer compared to reading from the process screen.
- The duration of the call depends, among other things, on the bus load and the AS.
- The synchronous call causes a higher communication load once, but the tags are not included in the tag screen and are therefore only updated once.

```
' VBS
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read(1)
```

Notes

Direct reading from the AS (synchronous call) is always useful if

- fast read/write procedures are to be synchronized,
- a value is read explicitly from the AS,
- or a registration in the screen should be deliberately avoided.

During the runtime, the processing of the actions is blocked and the duration of the call cannot be predicted. If multiple tags are read, the time is added together. Therefore, always use the asynchronous call if possible.

Reading tags with status handling

To ensure that a value is valid, a check can be performed after reading. This is done by checking the quality code.

In the following example, the tag "myWord" is read with status handling.

- After reading, the QualityCode is 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.
- If an error occurs during reading, the QualityCode is set to "BAD Out of Service".

```
' VBS
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
```

4.2.6 Writing Tags

Writing tags asynchronously without object reference

In asynchronous writing, the value is transferred to the tag management and the processing of the action is continued.

```
' VBS
HMIRuntime.Tags("Tag1").Write 6
```

Writing tags asynchronously with object reference

In asynchronous writing with an object reference, a local copy of the tag object is first created before processing of the action is continued.

```
' VBS
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 6
```

Referencing offers the advantage of being able to work with the tag object before writing. You can read the tag values, perform calculations, and write them again:

```
' VBS
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
objTag.Value = objTag.Value + 1
objTag.Write
```

Writing tags synchronously with object reference

In some cases, it must be ensured that the value has actually been written before the action is processed further.

This type of writing is realized by specifying the value 1 for the additional, optional parameters:

```
' VBS
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 6
objTag.Write ,1
```

Another way of writing this is:

```
' VBS
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 6,1
```

Notes

Please note that this call takes longer in comparison to the standard call. The duration depends, among other things, on the channel and the controller.

In contrast to asynchronous writing, where the tag values are cyclically updated by tag management, synchronous writing handles the value only once.

This write type corresponds to the SetTagXXXWait() call with ANSI-C.

Writing tags 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:

- After writing the value, the "LastError" property is checked.
- If the check was successful, i.e. the job was successfully completed, the status of the tags is checked.

Notes

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 contained in the "QualityCode" property after the read operation provides information about the status of the tags and, if necessary, indicates a failed controller connection.

In the following example, the tag "Tag1" is written with status handling:

- If an error occurs during writing, the error value and error description appear in the Global Script diagnostics window.
- Finally, the QualityCode is checked.
- If the QualityCode is not OK (0x80), this is displayed in the diagnostics window.

```
' VBS
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Write 6
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
```

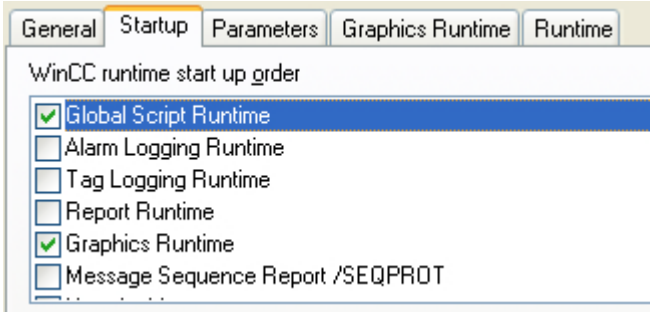
4.2.7 Global Actions in WinCC V7

In the "Global Script Editor", screen-independent scripts are configured.

To ensure that global scripts are executed in Runtime, Global Script Runtime must be included in the project's startup list:

8. Select "Properties" in the computer context menu in WinCC Explorer. The "Computer list properties" dialog opens.
9. Click the "Properties" button. The "Computer properties" dialog opens.
10. Select the "Startup" tab.
11. Activate Global Script Runtime.

Figure 4-5



12. Click "OK" to close the dialog.

4.2.8 Global Actions in WinCC Professional

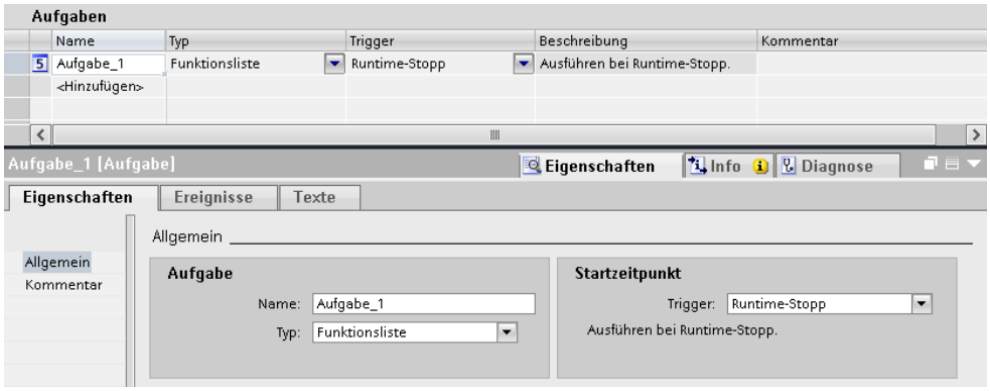
In the task planner, tasks are configured that are executed in the background independently of the screen. You create the tasks by linking system functions or scripts to a trigger. When the triggering event occurs, the linked functions are called.

Double-click "Scheduler" to open it in the project view.

The work area displays the scheduled tasks, which consist of

- the trigger (for example, stop Runtime) and
- the task type (for example, a function list).

Figure 4-6



4.3 Error Diagnostics

There are several options for diagnosing errors in VBS:

- Trace commands
  The targeted implementation of trace commands, e.g. for the output of tag values, enables the progress of actions and the procedures called in them to be traced.
- Error handling via the error object
  The Error object provides certain properties and methods for this purpose.
- Script debugger
  The "Microsoft Visual Studio 2008" script debugger is used to display errors in local and global scripts.
- Debugging with Visual Studio
  - Executing scripts step-by-step
  - Set and delete breakpoints
  - Setting and clearing bookmarks
  - Monitoring tags and objects
- Log files
  An access violation in C or VB scripts generates the "Script.log" file. The file "Script.log" is located in the folder "…\Siemens\WinCC\Diagnostics" of the WinCC installation path

Notes

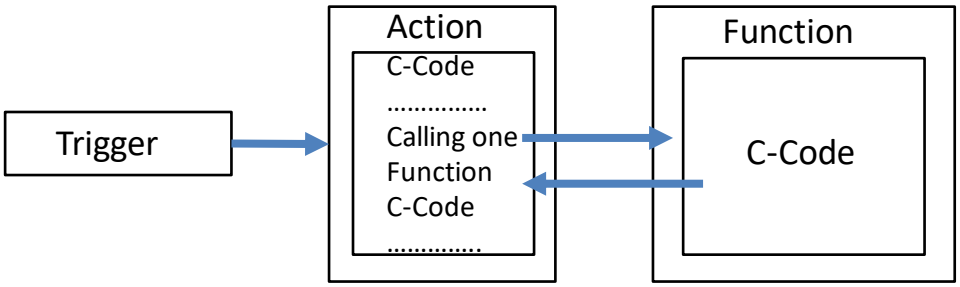
You can find a detailed description of error diagnostics in scripts in the documentation for the application example "Diagnostics of WinCC V7", section 4:

https://support.industry.siemens.com/cs/ww/en/view/109757865

5 ANSI-C

5.1 Interface Description

Figure 5-1



Actions are divided into:

- Global actions
- Local actions

Functions are divided into:

- Internal functions
- Standard functions
- Project functions

Notes You can find additional information on when to use which actions or functions – and in which path they are stored – in section 3.2.2.

5.1.1 Actions

- Actions are always activated by a trigger – namely a triggering event.
- You configure actions as properties of graphic objects, events that occur for a graphic object, or globally in the project.
- In actions, you can call code that is used several times in the form of functions.
- An action has no parameters.
- Actions in the Global Script can be protected against modification and viewing by means of a password.

5.1.2 Functions

- A function corresponds to a procedure in VBS.
- Functions are used to store code that you want to use at several places in your project configuration.
- 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.
- Functions can be created in WinCC with or without return values.

5.2 Integration in the User Project

5.2.1 Editor

WinCC provides the "Global Script" editor for designing, creating, and editing functions and actions. Global Script is started from the navigation window of WinCC Explorer.

Unicode support

You can set the suitable code page in the toolbar of the "Global Script" editor. This means that the system language no longer has to be changed with the Microsoft setting "Start > Settings > Control Panel > Regional and Language Options".

You can select "Dynamic: project setting" as the language setting for scripts. The C script is compiled in English. The code page of the centrally configured language is used for the strings in Runtime.

You can specify the project setting in the "Project Properties" dialog in the WinCC Explorer. You can select the following from a list of "Options" under "C scripts with 'Dynamic' language setting in Runtime":

- The respective WinCC Runtime language is set:
  The C script is executed in the WinCC Runtime language.
- Operating system language for non-Unicode programs:
  Select the language from the list.

The C script is executed with the code page setting of the operating system.

5.2.2 Creating and Editing 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.

Notes Internal functions cannot be created or edited.

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 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, therefore, remains clearer.

Writing function code

- The function code is written in the Edit window for the function. The programming language is ANSI-C.
- To make certain that the called function is recognized by 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 return value and the default name of the new function.
- Parameters can be passed by entering them in the subsequent brackets.
- The function code is entered between the curly brackets.
- The code of any project or standard function can call other functions:
  - Project functions,
  - Standard functions,
  - Internal functions, or
  - DLL functions

Internal functions

You can use any of the internal functions as part of your function code. You can find internal functions in the navigation window in the "Internal functions" group.

Procedure:

1. Place the cursor at the point at which the internal function is to be inserted.
2. In the navigation window, open the context menu for the internal function to be added
3. Select "Assigning parameters". The parameterization dialog contains one line for each parameter. In the "Value" column, enter the respective current parameter.
4. Enter the current value for each required parameter in the "Value" column.

5. Confirm your entries with "OK". The parameterized function is inserted in the edit window at the location of the cursor.

Standard and project functions

You can use all project and standard functions as part of the function code. 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.

- You can find the project functions in the navigation window in the "Project functions" group.
- You can find the standard functions in the navigation window in the "Standard functions" group.
- The project functions are entered in the header file "Ap_pbib.h".
- The standard functions are entered in the header file "Ap_glob.h".
- 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.

Procedure:

Place the cursor at the point at which the project or standard function is to be inserted.

1. In the navigation window, open the context menu for the function to be added.
2. Select "Assigning parameters". The parameterization dialog contains one line for each parameter. In the "Value" column, enter the respective current parameter.
3. Enter the current value for each required parameter in the "Value" column.
4. Confirm your entries with "OK"

Notes

- If the function does not require a parameter, it is added to the function code immediately without opening the Parameterization dialog.
- If you close the Parameterization dialog with "OK" without entering the current parameter value, the internal function is inserted with its default parameters. You can then set the parameterization in the Edit window at a later point.

Notes

You can find additional information on the relationships between actions and functions in section 5.1, "Interface Description".

5.2.3 Creating and Editing Actions

Actions can be configured in Global Script and in the Graphics Designer:

- Use Global Script to configure global actions which can be executed independently of the screen currently open.
- Graphics Designer is used to configure actions related to graphic objects which should be executed when the screen is opened in Runtime or when the configured trigger occurs.

A distinction is made between global and local actions:

- 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).
- In a Client–Server project, global actions are carried out on all computers in the project, whereas local ones are executed only on the computer to which they are assigned.
- For example, a global action can be used to perform a calculation on all computers in the project.
- A local action can be used, for example, to output a protocol to a server.

Editing action code

- 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.
- The return value can be changed and displayed in connection with GSCRuntime 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.

Notes You can find additional information on the relationships between actions and functions in section 5.1, "Interface Description".

5.2.4 Use of DLLs in Functions and Actions

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.

For example, to use the functions <function name 1> ... <function name n> from the DLL <name.dll> in a function or action, the following code must be added to the beginning of the declaration:

```
// ANSI-C
#pragma code("<Name>.dll")
<Typ des Returnwerts> <Funktionsname 1>(...);
<Typ des Returnwerts> <Funktionsname 2>(...);
.
.
.
<Typ des Returnwerts> <Funktionsname n>(...);
#pragma code()
```

Example:

```
// ANSI-C
#pragma code("kernel32.dll")
VOID GetLocalTime(LPSYSTEMTIME lpSystemTime);
#pragma code()

SYSTEMTIME st;
GetLocalTime(&st);
```

As an alternative to this procedure, you can also make the necessary additions in the "Apdefap.h" header file.

Notes

- When using your own DLLs in WinCC, the release version must be used, otherwise DLLs are loaded both as release and debug versions. Increasing the memory requirements.
- Structures of the DLL have to be set up using 1-Byte alignment.
- 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.

5.2.5 Reading Tags

Basically, a distinction is made between two types of tag updating:

Read from the tag screen (asynchronous call)

- When reading from the tag screen, the tag is registered and, from then on, cyclically requested from the automation system.
- The login cycle is dependent on the configured trigger.
- The value is read from the tag screen by WinCC.
- When a screen is deselected, the displayed tags are deregistered again.
- The call is faster compared to direct reading. However, the first call takes longer, since the value must first be read from the AS and registered.
- 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.

```
// ANSI-C
{
WORD wValue;
wValue = GetTagWord("gs_tag_word");
}
```


Direct reading from the AS (synchronous call)

- The tag is not registered cyclically, the value is requested from the AS one time only.
- The current value is read explicitly from the AS.
- The call takes longer compared to reading from the process screen.
- The duration of the call depends, among other things, on the bus load and the AS.
- The synchronous call causes a higher communication load once, but the tags are not included in the tag screen and are therefore only updated once.
- The function does not deliver any information on the status of the tags.

```
// ANSI-C
{
WORD wValue;
wValue = GetTagWordWait("gs_tag_word");
}
```

Notes

Direct reading from the AS (synchronous call) is always useful if

- fast read/write procedures are to be synchronized,
- a value is read explicitly from the AS,
- or a registration in the screen should be deliberately avoided.

During the runtime, the processing of the actions is blocked and the duration of the call cannot be predicted. If multiple tags are read, the time is added together. Therefore, always use the asynchronous call if possible.

Reading tags with status handling

The function GetTagXXXState has the same features as GetTagXXX, it also sends the function information on the state of the tags. Since the status is always delivered internally, there is no performance difference to GetTagXXX.

```
// ANSI-C
{
DWORD dwstate;
WORD wValue;
dwstate = 0xFFFFFFFF;
//Get the tag value
//dwstate is the tag state
wValue = GetTagWordState("gs_tag_word",&dwstate);
}
```

5.2.6 Writing Tags

Writing tags asynchronously

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 has the following properties:

- The call is fast.
- The caller does not know when the value is actually written.
- The function provides no information on the status of the write job.

```
// ANSI-C
{
//Set the tag to 50
SetTagWord("gs_tag_word",50);
}
```

Writing tags synchronously

In some cases, it must be ensured that the value has actually been written before the action is processed further.

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 has the following properties:

- 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 status of the write job.

```
// ANSI-C
{
//Set the tag to 50
SetTagWordWait("gs_tag_word",50);
}
```

Notes

Please note that this call takes longer in comparison to the standard call. The duration is also dependent on the channel and AS, amongst other things.

In contrast to asynchronous writing, where the tag values are cyclically updated by tag management, synchronous writing handles the value only once.

Writing tags 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:

- After writing the value, the "LastError" property is checked.
- If the check was successful, i.e. the job was successfully completed, the status of the tags is checked.

Notes

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 "QualityCode" property after the read process provides an indication of the tag status and, if necessary, makes reference to a failed AS connection.

The call has the following properties:

- The call is fast.
- The caller does not know when the value is actually written.
- The function provides information about the status of the write job.

```
// ANSI-C
DWORD dwstate;
//Load dwState with default values
dwstate = 0xFFFFFFFF;
//Set the tag to 50
//dwstate is the tag state
SetTagWordState ("gs_tag_word",50,&dwstate);
```

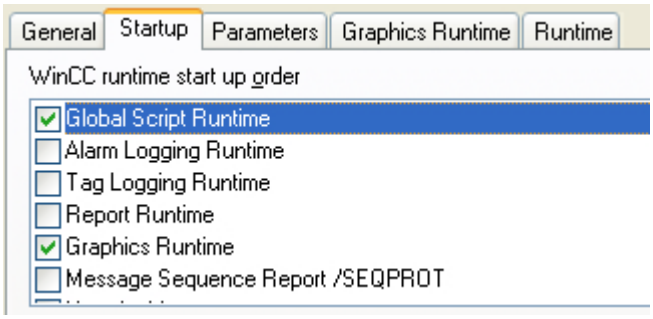
5.2.7 Global actions in WinCC

In the "Global Script Editor", screen-independent scripts are configured.

To ensure that global scripts are executed in Runtime, Global Script Runtime must be included in the project's startup list:

1. Select "Properties" in the computer context menu in WinCC Explorer. The "Computer list properties" dialog opens.
2. Click the "Properties" button. The "Computer properties" dialog opens.
3. Select the "Startup" tab.
4. Activate Global Script Runtime.

Figure 5-2



5. Click "OK" to close the dialog.

5.2.8 Global Actions in WinCC Professional

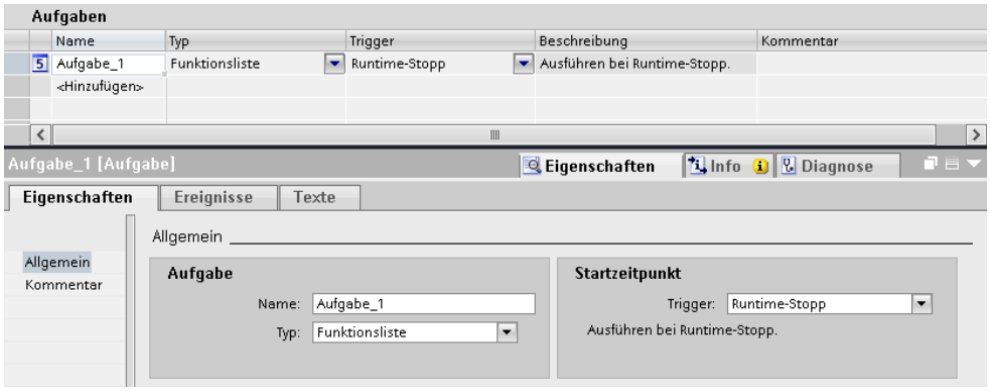
In the task planner, tasks are configured that are executed in the background independently of the screen. You create the tasks by linking system functions or scripts to a trigger. When the triggering event occurs, the linked functions are called.

Double-click "Scheduler" to open it in the project view.

The work area displays the scheduled tasks, which consist of

- the trigger (for example, stop Runtime) and
- the task type (for example, a function list).

Figure 5-3



5.3 Error Diagnostics

Several options are available for error diagnosis in ANSI-C:

- printf command
  Through the targeted use of printf commands, for example, to issue tag values, the sequence of actions and the functions called in it can be tracked.
- Diagnostics tags
  WinCC offers a number of system tags, including three diagnostic tags, for the status display of script processing.
  - @SCRIPT_COUNT_TAGS
    This tag contains the current number of script-requested tags.
  - @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.
- Application diagnostics (ApDiag)
  The "ApDiag.exe" diagnostic tool supports the analysis of errors and performance issues.
- Performance analysis
  A targeted performance analysis can be carried out through the targeted extension of the standard "OnTime" function.
- Log files
  An access violation in C or VB scripts generates the "Script.log" file. The file "Script.log" is located in the folder "…\Siemens\WinCC\Diagnostics" of the WinCC installation path
- System messages
- Global Script Inspector
  The "CCScriptInspector.exe" diagnostic tool examines the source code of the global scripts and indicates potential errors at runtime.

Notes

You can find a detailed description of error diagnostics in scripts in the documentation for the application example "Diagnostics of WinCC V7", section 4:

https://support.industry.siemens.com/cs/ww/en/view/109757865 .

6 VBA

The Graphics Designer provides a VBA Editor that allows you to automate the project configuration of screens. The VBA Editor is identical to that in the products of the Microsoft Office family.

The VBA Editor included in the Graphics Designer supports you in these tasks:

- Creating user-defined menus and toolbars
- Creating and processing standard objects, smart objects, and Windows objects
- Dynamizing properties of screens and objects
- Configuring actions for screens and objects
- Accessing products that support VBA (e.g. products of the MS Office family)

6.1 Interface Description

You can organize the VBA code of your WinCC project in the VBA Editor. This is where you specify whether the VBA code is to be available in only one screen, 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
  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.
- Project-specific VBA Code
  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.
- Screen-specific VBA code
  Refers to VBA code that you write to the document "This Document" relating to the corresponding screen in the VBA Editor. This VBA code is saved as a PDL file together with the screen.

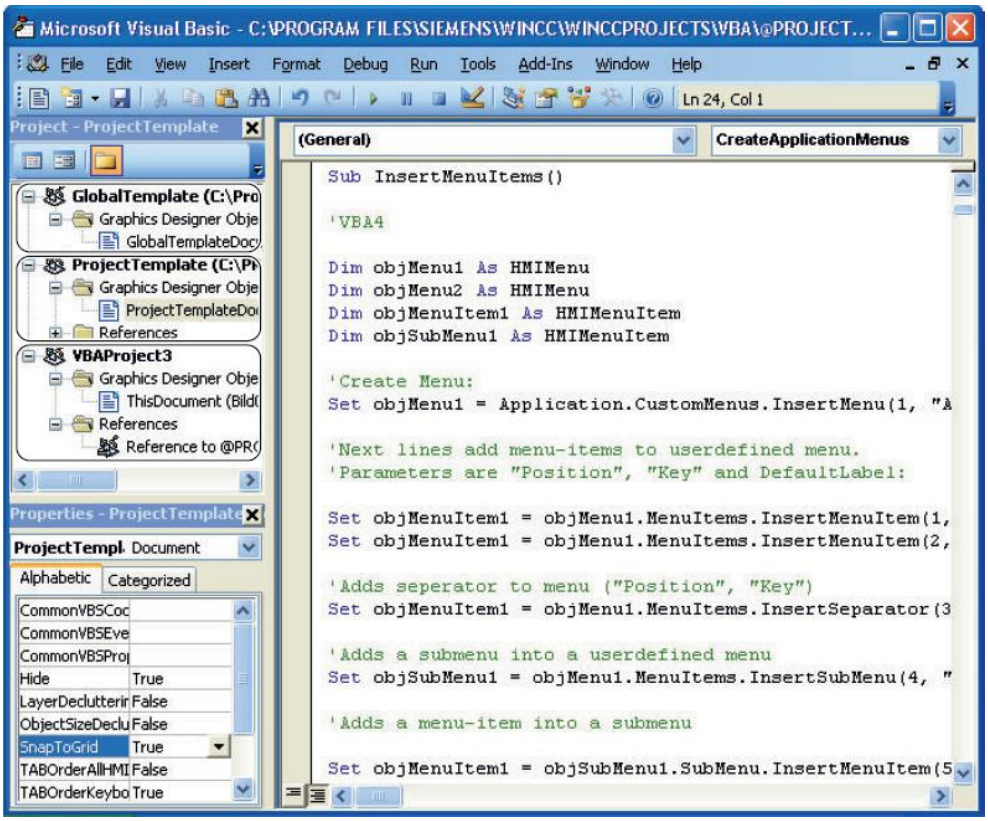
Notes For more information on the respective dependencies and available functions and procedures in screen-specific and project-specific VBA code, see section 3.2.3 .

6.2 Editor

To start the VBA Editor in the Graphics Designer, press <Alt+F11> or choose "Extras > Macros > Visual Basic Editor". If you have not yet opened a screen in the Graphics Designer, you can only edit the global or project-specific VBA code.

The global and project-specific data and all open screens are displayed in the VBA Editor's Project Explorer:

Figure 6-1



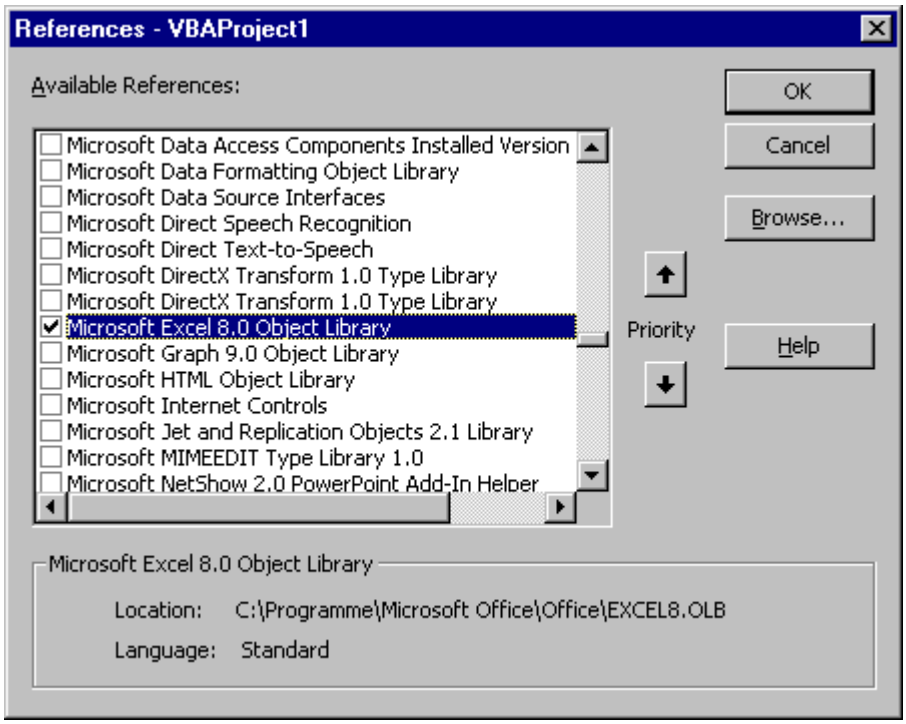
6.3 Access to External Applications

You can use VBA to access programs that 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:

Figure 6-2



Notes

You can find additional information on accessing third-party applications with VBA in the manual "WinCC V7.5 SP1: Scripting" – "Access to Third-Party Applications with VBA":

https://support.industry.siemens.com/cs/ww/en/view/109773213/45227085963

6.4 Error Diagnostics

You can test your VBA scripts in Runtime with the VBA Editor's debugger. You can find additional information in the help section of the VBA Editor.

7 ODK – Runtime API

The ODK (WinCC/Open Development Kit) is a WinCC option for using the open programming interfaces, which allows access to data and functions of the WinCC project configuration and the WinCC Runtime system.

The Runtime API is a port of the ODK to WinCC Runtime Professional, but with reduced functionality. For example, the ODK functions for CS are not included in WinCC Professional.

Notes

- The ODK documentation for WinCC V7 is subject to a fee.
- The Runtime API, as well as the associated documentation, is part of the scope of supply of WinCC Runtime Professional.

With the ODK, you get:

- the documentation for the API functions of WinCC in CHM files. In addition to the description of the API functions, notes explaining the respective examples have been included.
- Samples in C and C++, which demonstrate how simple it is to use API calls. All examples are provided for Visual Studio 2010 and newer. The samples are organized in a number of "Visual C++" projects:
  - A main project, to which the help files refer. This also includes the executable program, which you can generate yourself at any time.
  - Various projects with samples of the API calls. Some of these projects have in part emerged from the test environment of the software engineers; they do not only contain API calls, they also show how to build simple interactive programs using the MFC classes.
- Various samples with C scripts. Only a few scripts are supplied, as the scripts belong to the standard WinCC scope of delivery and scripts can also be generated from the samples.

The Runtime API documentation is part of the Online Help section for WinCC Runtime Professional.

API functions

API functions are configuration and Runtime functions, such as:

- MSRTCreateMsg: Creates a message
- DMGetValue: Determines the value of a tag
- PDLRTSetProp: Sets the object properties of a screen

API functions can be used in the following circumstances:

- In WinCC, e.g. in Global Script or within the scope of C actions in the Graphics Designer.
- In Windows applications in the C programming language.

8 Appendix

8.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:
support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:
www.siemens.com/industry/supportrequest

SITRAIN – Training for Industry

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:
www.siemens.com/sitrain

Notes

The following SITRAIN courses provide practical knowledge about the products used in this application example:

- WinCC V7 Fundamentals course/System course
  Article ID: 109758633
- WinCC Professional Fundamentals course/System course
  Article ID: 109758618

Practical knowledge of the application possibilities of Visual Basic scripts is taught in this SITRAIN course (only available in german language):

- Visual Basic Script in der SIMATIC-Welt (de)

Practical knowledge of the application possibilities of C scripts is taught in this SITRAIN course:

- ANSI-C in der SIMATIC-Welt, Einführung (de)
- ANSI-C in the SIMATIC World, Introduction (en)

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:
support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:
support.industry.siemens.com/cs/ww/en/sc/2067

8.2 Links and literature

Table 8-1

No.	Subject
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the article page of the application example https://support.industry.siemens.com/cs/ww/en/view/109773206
\3\	

8.3 Change documentation

Table 8-2

Version	Date	Change
V1.0	07/2020	First version