

# Display of Process Values in a 3D-Grid in WinCC Runtime

SIMATIC WinCC

Application Description • August 2011

Applications & Tools

Answers for industry.

**SIEMENS**

## **Industry Automation and Drive Technologies Service & Support Portal**

This document is taken from the Service Portal of Siemens AG, Industry Automation and Drive Technologies. The following link takes you directly to the download page of this document.

<http://support.automation.siemens.com/WW/view/en/49492528>

If you have any questions concerning this document please e-mail us to the following address:

[online-support.automation@siemens.com](mailto:online-support.automation@siemens.com)

# SIEMENS

## SIMATIC WinCC 3D-Grid

Automation Problem

1

Automation Solution

2

Functional Mechanisms  
of this Application

3

Configuration

4

Installation

5

Commissioning of the  
Application

6

Operation of the Sample  
Project of the Application

7

References

8

History

9

## Warranty and Liability

### Note

The application examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The application examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are correctly used. These application examples do not relieve you of the responsibility of safely and professionally using, installing, operating and servicing equipment. When using these application examples, you recognize that Siemens cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these application examples at any time without prior notice. If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – then the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc. described in this application example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above provisions do not imply a change in the burden of proof to your detriment.

It is not permissible to transfer or copy these application examples or excerpts of them without first having prior authorization from Siemens Industry Sector in writing.

# Foreword

## Objective of the application

The representation and visualization of processes is one of the main tasks of a user interface between human and machine (HMI).

Especially during ongoing production or continuous work processes, it is important that the data of a machine be visualized so that the operator can identify, read and understand the information with a few glances.

# Table of Contents

<b>Warranty and Liability .....</b>	<b>4</b>
<b>Foreword.....</b>	<b>5</b>
<b>1 Automation Problem .....</b>	<b>7</b>
1.1 Overview.....	7
1.2 Description of the automation problem .....	7
<b>2 Automation Solution .....</b>	<b>8</b>
2.1 Overview of the overall solution .....	8
2.2 Core functionality of the VBA macro .....	10
2.3 Core functionality of the VBScripts in Runtime .....	11
2.4 Software components used.....	11
2.5 Performance data.....	12
<b>3 Functional Mechanisms of this Application .....</b>	<b>13</b>
3.1 Functionality of the VBA macro .....	13
3.2 Functionality of the VBScripts .....	14
3.2.1 Differences between the script files .....	14
3.2.2 Global Script action (3D_Grid_GSC_Action_AktuellesDatum_Uhrzeit).....	15
3.2.3 Picture script (3D_Grid_Picture_Script_AktuellesDatum_Uhrzeit) .....	17
<b>4 Configuration (for your own project).....</b>	<b>20</b>
<b>5 Installation.....</b>	<b>22</b>
<b>6 Commissioning of the Application .....</b>	<b>23</b>
6.1 Preparation.....	23
6.2 Creating a 3D-Grid in the Graphics Designer .....	23
6.3 Adapting the VBScripts to the purpose .....	24
<b>7 Operation of the Sample Project of the Application .....</b>	<b>27</b>
7.1 Overview.....	27
7.2 "Classic" process screen .....	27
7.3 "Application example" process screen .....	28
<b>8 References .....</b>	<b>29</b>
8.1 References .....	29
8.2 Internet links .....	29
<b>9 History.....</b>	<b>29</b>

# 1 Automation Problem

## 1.1 Overview

A frequent requirement is to clearly display several comparable measured values of a plant.

The aim of this application example is to show you a way to meet this requirement.

## 1.2 Description of the automation problem

The temperature of ten furnaces of a plant is to be graphically represented. The measured values are acquired at hourly intervals.

At the end of the production shift, the hourly temperature values of all ten furnaces are to be displayed at a glance.

## 2 Automation Solution

### 2.1 Overview of the overall solution

#### Display of several measured values with time characteristics

Using different forms of diagrams, process values can be displayed so that the relationships between the values become clear. For this reason, certain diagram types are often only suitable for specific values.

If several values are to be displayed with dependencies, it is advantageous to use a representation that is similar to a coordinate system as known from mathematics since it provides several axes to represent the values.

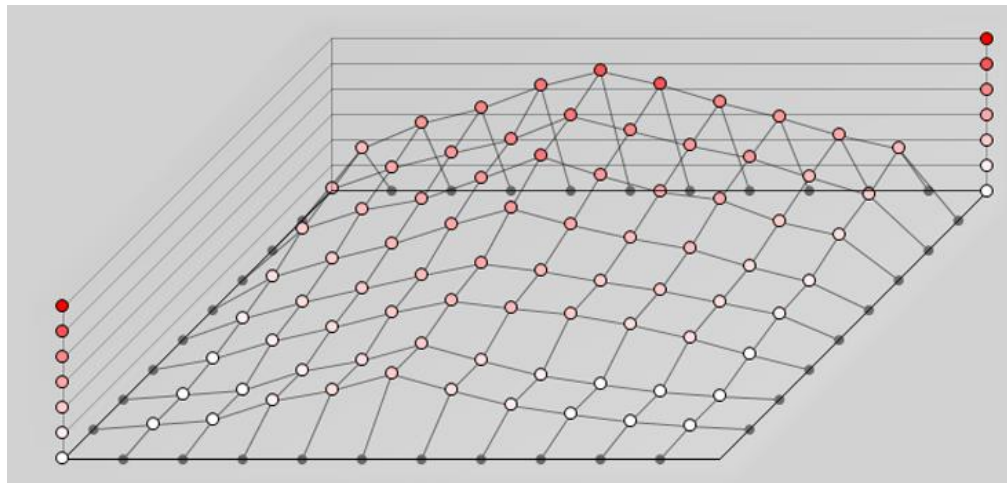
The 3D-Grid offers you the option to clearly display several measured values with various dependencies.

To give a possible practical example, it provides the operator with a direct comparison between the hourly values of all ten furnaces of the plant.

The application describes the use of a macro for automatic configuration of a 3D-Grid and the modification and use of the necessary scripts (VBS).

**NOTICE** The scripts generated by the macro are default scripts that include the basic principle of operation. You have to adapt these scripts to your requirements and then check them for correct functionality.

Figure 2-1

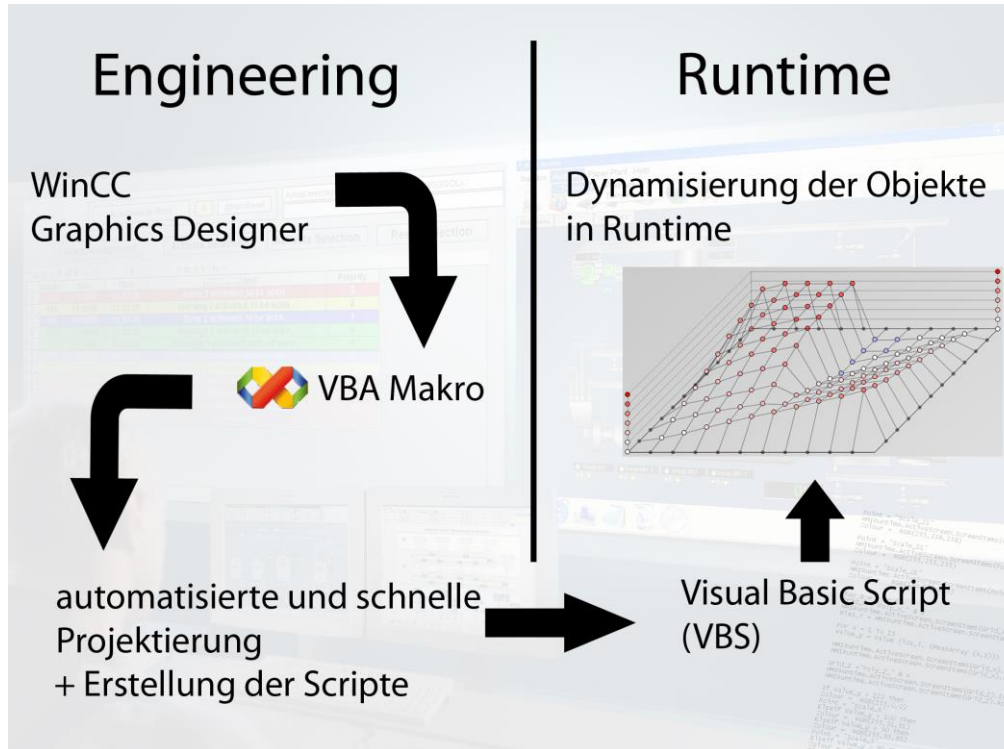




### Diagrammatic representation

The diagrammatic representation below shows the correlation between the most important components of the solution:

Figure 2-2



### Structure

As can be seen in the figure, the application consists of two main components:

- VBA macro (engineering side)
  1. Configuring the objects
  2. Creating the VBScripts
  
- VBScripts (Runtime side)
  1. Dynamizing the objects in Runtime

### Scope

This application does not include a description of

- the creation of VBA macros in the Graphics Designer.
- the dynamization of objects in WinCC using VBS.
- VBS and VBA basics.
- the configuration with SIMATIC WinCC.

Basic knowledge of these topics is required.

**Required knowledge**

Basic knowledge of the VBS and VBA script languages is required. Knowledge of WinCC is also necessary.

**NOTICE** This application is only suitable for users who already have experience with (VBS) scripting in a WinCC environment.

If the scripts are negligently modified, this may have a strong impact on the performance of your system and even cause the application to crash.

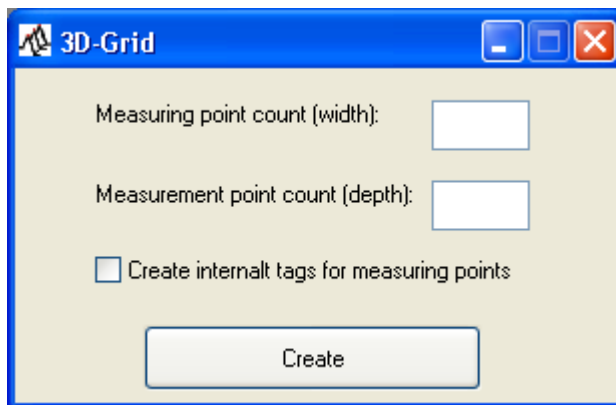
## 2.2 Core functionality of the VBA macro

**Definition**

In the software environment, macros are used to supplement and thus automate frequently used operational steps as functions in an already existing program. Time saving and lower susceptibility to faults make work easier for the user.

**Overview and description of the user interface**

Figure 2-3



Two input fields are available. They enable you to define the width and depth of the 3D-Grid.

In addition, you can define whether internal tags are to be created for the measuring points.

**Sequence of the core functionality**

Within the macro, two main tasks are performed.

- Configuring the objects.
- Creating the VBScripts as text files.

**Note**

How quickly the macro is created depends on the computing power of your PC. It may take a few moments until all objects are created.

**Advantages of this solution**

The solution presented here offers you the following advantages:

- Automatic configuration of the grid, which reduces the configuration overhead and thus results in significant savings in time and costs.
- Extension of WinCC by another diagram type, which enables you to implement your individual requirements.
- Additionally lower susceptibility to faults due to the automatic script generation.

**2.3 Core functionality of the VBScripts in Runtime**

The scripts that are automatically generated when running the macro perform two tasks:

- Reading the current values out of the tags (measuring points) and then saving them to internal tags.
- Dynamizing the 3D-Grid depending on the values from the internal tags.

**2.4 Software components used**

The application was created with the following components:

**Hardware components**

Table 2-1

Component	Qty.	MLFB/order number	Note
SIMATIC Rack PC	1	6AG4104-0DA14-1BX0	

**Software components**

Table 2-2

Component	Qty.	MLFB/order number	Note
WinCC V7.0 SP2	1	6AV63.1-....7-0...	To ensure that the application runs correctly, you should also use this version.

### Sample files and projects

The following list contains all files and projects that are used in this example.

Table 2-3

Component	Note
3D-Grid_Demo_Project_V_1_0.zip	Zipped WinCC sample project to show the functionality.
WinCC_3D_Diagramm_en.pdf	This document.
3D-Grid_Makro_EN.zip	Zipped DLL file (VBA macro) to create your own 3D-Grids. (German and English)

## 2.5 Performance data

### Hardware

The computer hardware on which you run the application should at least meet the recommended hardware requirements from the WinCC Information System. (Installation Notes > Installation Requirements > [Hardware Requirements for the Installation](#))

### Software

On the software side, there are two factors that limit the size and dimensioning of this application.

The size of the process screen in the WinCC Graphics Designer decides how many measuring points (width) and how many measuring times (depth) you can visualize in one screen.

The number of available interface tags (PowerTags) decides how many measuring points you can display when the received values come directly from an external device (e.g., PLC).

### Note

To efficiently use external interface tags, the global script (in the sample project) is structured so that the measured values of the external tags are acquired at a specific cycle (every minute) and then written to license-free internal tags. This method saves you external interface tags (requiring a license).

## 3 Functional Mechanisms of this Application

### General overview

This chapter provides you with information on the basic principle of operation of the VBA macro and the VBScripts. Since the VBA macro is a closed application (dll) and you as a user have no influence on the principle of operation, this chapter focuses on explaining the used scripts.

### 3.1 Functionality of the VBA macro

As already explained in the [previous chapter](#), the user of the application can only influence the width and depth of the later 3D-Grid.

(The maximum height of the grid is defined by the maximum value of the tags and the height of the process screen (pdl).)

Using these two values (width, depth), the program calculates, in the background, the position of all objects that are created during the configuration.

The distance between the measuring points (width) and the distance between the measuring times (depth) cannot be influenced. However, the distance was selected so that the values can be clearly displayed, while still making optimum use of the space.

### Program details

You can separately download the VBA macro of this application.

The program code of a dll cannot be changed or influenced by the user. This ensures faultless functioning of the macro.

The integration of this file into the operating system is explained in a [following chapter](#).

A description of registering a dll in the operating system is also available in the WinCC Information System: Working with WinCC > VBA for Automated Configuration > Add-Ins > [Integrating Add-Ins](#).

## 3.2 Functionality of the VBScripts

### 3.2.1 Differences between the script files

When running the VBA macro, three text files are generated in the “GraCS” folder of the project:

*3D\_Grid\_GSC\_Action\_“AktuellesDatum\_Uhrzeit”.txt*

*3D\_Grid\_Picture\_Script\_“AktuellesDatum\_Uhrzeit”.txt*

*3D\_Grid\_CLASSIC\_“AktuellesDatum\_Uhrzeit”.txt*

In practice, you will mostly need the scripts of the first two files. This is a script that is triggered directly in the screen

(*3D\_Grid\_Picture\_Script\_AktuellesDatum\_Uhrzeit.txt*). The second file is a Global Script action that reads the tag values as already mentioned in [chapter 2.5](#).

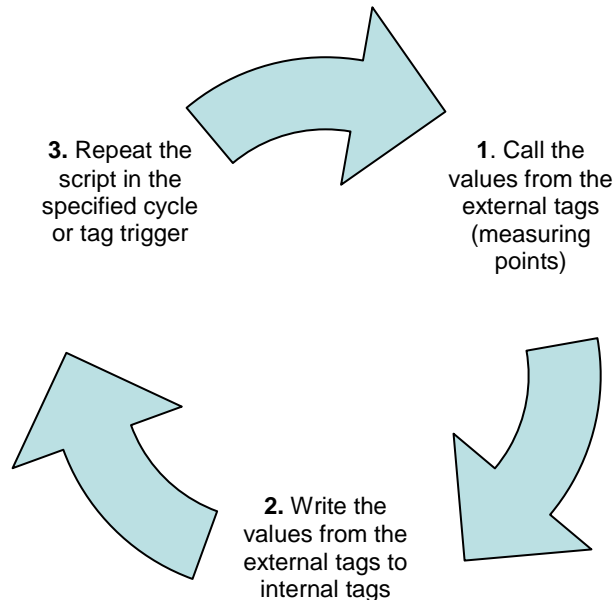
(*3D\_Grid\_GSC\_Action\_“AktuellesDatum\_Uhrzeit”.txt*).

The third text file (*3D\_Grid\_CLASSIC\_“AktuellesDatum\_Uhrzeit”.txt*) is generated for the case that you want to make major changes to the functionality of the scripts yourself. This file contains a script that (nearly) combines the functionality of the two other files. (The mechanism for writing the tag values from external tags (PowerTags) to internal tags is missing).

If you want to implement a similar use case, as is the case in the sample project for this application, you will not necessarily need this file.

### 3.2.2 Global Script action (3D\_Grid\_GSC\_Action\_AktuellesDatum\_Uhrzeit)

#### Principle of operation



As you can see in the diagram, the Global Script action is a loop that is responsible for ensuring that the current measured values are written to internal tags for “buffering”.

This requires that there be one internal tag for each measured value to be acquired. (For a 3D-Grid with ten measuring points and eight measuring times, this would mean that 80 internal tags have to be created)

#### Note

If you want to create a larger 3D-Grid, a large number of tags will be needed. To simplify the creation of these tags, it is useful to work with structure types/structure tags. In the sample project for this application, you will find a configuration with structure types/structure tags. The tag names used by default in the script are also designed for use with structure tags.

#### Specifying the tag name

The tag names of the external and internal tags are predefined in the script. When using tag names that differ from the ones in the script, you have to adjust them in the script.

#### Generally, the following applies:

By default, the tag names for the internal (storage) tags are composed of the “Poly\_” string, the serial number “idx\_grid”, the “.Wert\_” string and the serial number “idx\_Tag”. (The “idx\_grid” tag represents the index for the current measuring time (depth), while “idx\_Tag” is the index for the current measuring point (width).)

#### Example:

For a tag named "Poly\_4.Wert\_3", "Poly\_4" stands for the measuring time (depth) and, at the same time, represents the name of the structure tag (Poly\_4). "Wert\_3", however, stands for the measuring point (width) and is automatically assigned by the structure type. The name can thus always be used to identify the "measuring point" in the grid.

The tag names for the measuring points (possibly external tag) is composed of the "Grid\_Tag\_Poly\_" string and a serial number "idx\_Tag".

When you create a 3D-Grid with ten measuring points while the creation of internal tags for measuring points is enabled, ten tags named "Grid\_Tag\_Poly\_1" to "Grid\_Tag\_Poly\_10" will be created.

Figure 3-1

```
For idx_Tag = 1 To 10
    Set sTag = HMIRuntime.Tags("Poly_" & idx_grid.value & ".Wert_" & idx_Tag)
    Set iTag = HMIRuntime.Tags("Grid_Tag_Poly_" & idx_Tag)
    iTag.Read

    sTag.value = iTag.value
    sTag.Write

    HMIRuntime.Trace sTag.value & vbNewLine
Next
```

In lines two and three of the script section, you can see where the tag names are defined in the script.

"sTag" designates the structure tag (tag for buffering the value), while "iTag" represents the internal (measuring point) tag or also the external (measuring point) tag.

#### Note

In the engineering phase, it is advisable, as shown in the figure, to integrate trace outputs into the script code. In the event of an error, this facilitates diagnostics.

#### Limiting the cycles to available tags

Since the tag names are incremented after each cycle to write to the next "set of tags" during the next cycle, it must be ensured that this happens only while tags (measuring times) are available.

For a 3D-Grid with eight measuring times, this means that after the eighth cycle, the system must return to the first "set of tags" (first measuring time).



Figure 3-2

```

If idx_grid.value < 8 Then
    idx_grid.value = idx_grid.value + 1
    idx_grid.Write
Else
    idx_grid.value = 1
    idx_grid.Write
End If

```

As shown in the figure of the script code, the “idx\_grid” tag is checked during each cycle. If the value is less than “8”, the tag will be incremented by “1”. If this is not the case, the tag will be reset to “1”. An internal WinCC tag is used, which was configured with the start value “1”.

### 3.2.3 Picture script (3D\_Grid\_Picture\_Script\_AktuellesDatum\_Uhrzeit)

#### Overview

The script that is triggered directly in the process screen of the 3D-Grid can be divided into five sections:

- Measuring point assignment
- Reading in the tag values
- Coloring of the scale
- Position change of the measured values
- Coloring of the measured values

All sections are described below in greater detail.

#### Measuring point assignment

The first section of the script defines to which measuring point a measured value is assigned. Depending on the use case, the assignment can be flexibly changed.

Figure 3-3

```

MeasArray (1,1) = 1
MeasArray (1,2) = 1
MeasArray (1,3) = 1
MeasArray (1,4) = 1
MeasArray (1,5) = 1
MeasArray (1,6) = 1
MeasArray (1,7) = 1
MeasArray (1,8) = 1

```

The above figure shows the assignment for the measuring times of the first measuring point. The first number in brackets stands for the measuring point, the second one represents the measuring time. All eight measuring times were assigned to measuring point “1” (last number).

#### Reading in the tag values

The purpose of this section is to read in the values of all internal tags that were used as buffers to an array within the script.

#### Coloring of the scale

For better visualization, the side of the 3D-Grid has two scales that limit the grid height.

To provide the operator with an overview of how the color of the measured value changes, the points of the scale are colored in this part of the script. Change the RGB color code to adapt them to your requirements.

Figure 3-4

```
Colour = RGB(255,0,0)
Point = "Scale_6"
HMIRuntime.ActiveScreen.ScreenItems(Point).BackColor = Colour
```

#### Position change of the measured values

This section in the script represents the actual grid functionality. The "Y position" ("ActualPointTop" property) of all measured values is redetermined in a loop depending on the stored value in the tags.

Figure 3-5

```
For x = 1 To 10
Value_y = Value ( idx_i , (MeasArray (x,z)))

HMIRuntime.ActiveScreen.ScreenItems(Grid_X).Index = x + 1
HMIRuntime.ActiveScreen.ScreenItems(Grid_X).ActualPointTop = Bias_Y - Value_y

Grid_Z ="Poly_Z_" & x
HMIRuntime.ActiveScreen.ScreenItems(Grid_Z).Index = z + 1
HMIRuntime.ActiveScreen.ScreenItems(Grid_Z).ActualPointTop = Bias_Y - Value_y
```

#### Coloring of the measured values

In the last major section of the script, a query is used to check the current value of the measuring point. The color of the measuring point is then adjusted depending on this value.

Figure 3-6

```
If Value_y > 110 Then
Colour = RGB(255,0,0)
Point = "Scale_6"
Elseif Value_y > 100 Then
Colour = RGB(255,51,51)
Elseif Value_y > 90 Then
Colour = RGB(255,85,85)
Point = "Scale 5"
```

By adapting the script, you can define when a measured value is assigned which color.

Change the "RGB value" to adjust the color. Change the query to redefine the time when which color is displayed.

**Note**

To ensure that the operator still identifies the status of a value at first glance, you should continue to use “red colors” only for critical values.

## 4 Configuration (for your own project)

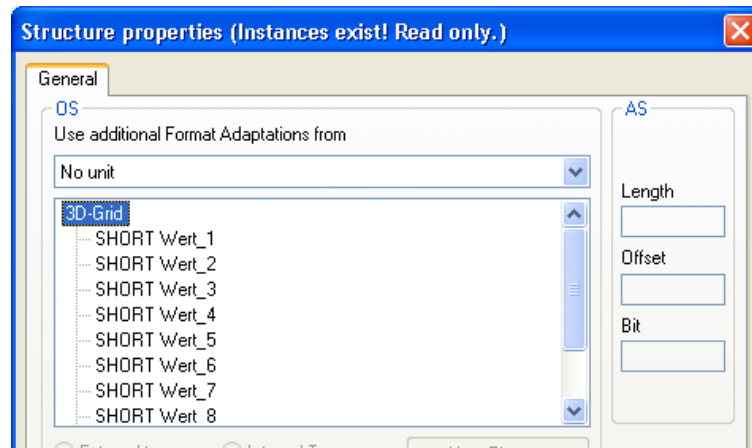
### Configuration of the tag structure

A structure must be created, which includes all associated measuring points for a measuring time.

When creating a 3D-Grid with ten measuring points and eight measuring times, you have to create a structure that contains ten internal tags.

When using the default tag names from the scripts, you have to name the tags with the "Wert\_1" to "Wert\_10" string.

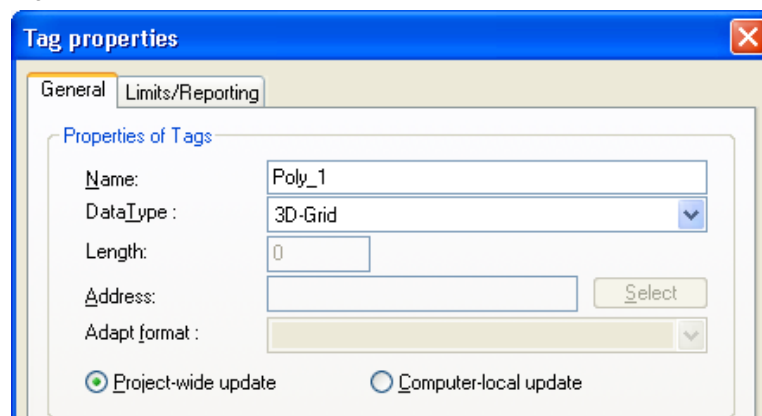
Figure 4-1



Then create a tag of the type of your structure for the first measuring time. Name this tag, for the first measuring point, "Poly\_1".

Ten tags are generated, which have the name structure that is used in the script. You have to perform this step for every measuring time (depth).

Figure 4-2

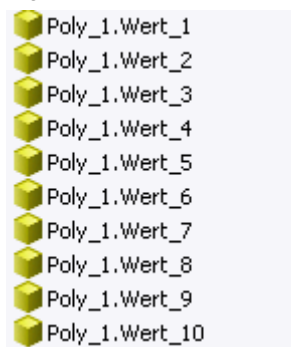


**Note**

The number of internal tags of the structure must match the number of measuring points. The number of measuring times specifies how often you have to create an internal tag of this structure type.

When you have generated a tag of the type of the previously created structure, you can see in Tag Management that the tags have the same name structure as defined in the script.

Figure 4-3



## 5 Installation

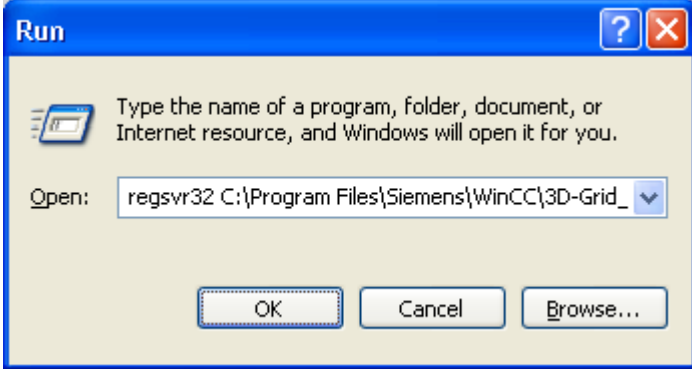
### Installation/registration of the VBA macro

To be able to use the application, you have to register the VBA macro in the operating system. This step is only necessary if you want to create a new 3D-Grid. If you have a project that includes a grid that has already been completed, this step is not necessary.

### Registering the VBA macro in the operating system

This chapter describes the procedure for registering the “dll” in the operating system.

Table 5-1

No.	Action
1.	Download the program library (dll) for this application (3D-Grid_Makro.zip).
2.	Unzip the file and copy it to any folder. (It is recommended to use, for example, the WinCC installation folder or a Windows system folder.)
3.	Click on “Start > Run” in the operating system.
4.	Enter the “regsvr32” command and the file path to which you have copied the file.  Example: 
5.	If the dll was registered correctly, a message indicating successful registration will be displayed on the screen.  If an error message appears, it is frequently caused by an incorrect file path or inadequate authorization in the operating system.

## 6 Commissioning of the Application

### 6.1 Preparation

To commission and use the application, the previous step “Registering the VBA macro” must have been successfully performed.

Before you create a 3D-Grid and start editing the scripts, you should think very carefully about what you want to display with the grid and which relevant conditions exist in your project.

### 6.2 Creating a 3D-Grid in the Graphics Designer

Table 6-1

No.	Action	Remark
1	Start the WinCC Graphics Designer and open a screen (pdl).	
2	In the menu bar, you find a new button labeled “3D-Grid”. Click on this button and select the “Create” menu item. The macro user interface opens.	If this is not the case, go to “Options > Add-In Manager”. Here you can set when the macro is loaded and thus displayed in the menu bar.
3	Define the width and depth of the 3D-Grid. If necessary, check the “Create internal tags for measuring points” option (the width and depth of the 3D-Grid result from the number of measuring points (width) and the number of measuring times (depth)).	The option to create internal tags is intended for the visualization of values that are not from an external device.
4	Select the “Create” button. The required objects are then automatically configured. Once the macro user interface disappears, the 3D-Grid, including the scripts, has been fully created.	The objects are always generated at the lower screen edge. This avoids that objects are placed outside the screen edge too quickly.
5	With Windows Explorer, open the GraCS folder of the project in which you have created the 3D-Grid.	
6	Depending on the specific use case you want to implement, open the <i>3D_Grid_CLASSIC_“AktuellesDatum_Uhrzeit”.txt</i> text file or the two files <i>3D_Grid_Picture_Script_“AktuellesDatum_Uhrzeit”.txt</i> and <i>3D_Grid_GSC_Action_“AktuellesDatum_Uhrzeit”.txt</i> .	
7	When using the script from the <i>3D_Grid_CLASSIC_“AktuellesDatum_Uhrzeit”.txt</i> file, you have to connect this script to a VBS action that is in the same screen as the 3D-Grid. When using the other two files, create a Global Script action for the script in the <i>3D_Grid_GSC_Action_“AktuellesDatum_Uhrzeit”.txt</i> file and copy the script from the file to this action. You then have to define a trigger for this action. Save the action. Copy the script from the <i>3D_Grid_Picture_Script_“AktuellesDatum_Uhrzeit”.txt</i> file also to a VBS action in the screen in which the 3D-Grid is located.	For the script of the <i>3D_Grid_CLASSIC_“AktuellesDatum_Uhrzeit”.txt</i> file, you can also create a VBS project module. This project module is then called in the screen. This configuration can be found in the sample project for this application in the “Classic” screen.

No.	Action	Remark
8	You can then run the script in the screen of the 3D-Grid, even if the used tags do not yet contain values.	The prerequisite is that the tags necessary for the script were created and that they have the correct names. When using the variant with one script, it is sufficient to check the "Create internal tags for measuring points" option. When using the variant with two scripts, please follow the instructions for creating tags in <a href="#">this chapter</a> .

## 6.3 Adapting the VBScripts to the purpose

### Overview

Only in very few cases will it be possible to apply the scripts generated by the macro to a project without modifications. This chapter shows you where to intervene in the scripts to achieve the desired "effect".

### Value acquisition

In the sample project, you will find a configuration in which the value of the measurement tag is acquired every minute. This cycle is solely defined by the trigger of the Global Script action. If you change this trigger to five minutes, the values will be acquired in a cycle of five minutes and "archived" in the internal tags.

### Range of values

Depending on which values you acquire, the range of values must/should be adjusted. In general, you can use each range of values in conjunction with the 3D-Grid; however, you should always adjust the coloring of the points to the range of values. A range of values from 0 to 100 is used in the sample project.

Figure 6-1

```

If Value_y > 110 Then

Elseif Value_y > 100 Then

Elseif Value_y > 90 Then

```



Range of values and color gamut can be extended by adding additional queries.

**Note**

The grid generated by the macro is optimized for a range of values from 0-100. When using a range of values from 0-200, you should also adjust the scale on the edge of the grid to the range of values to provide a clearer overview. In this case, the objects have to be manually copied or inserted. (Continue to name the scale points as in the existing points.)

**Assignment matrix**

As already described in "[Functionality of the VBScripts](#)", you can flexibly define the assignment of the measured values, as in a matrix.

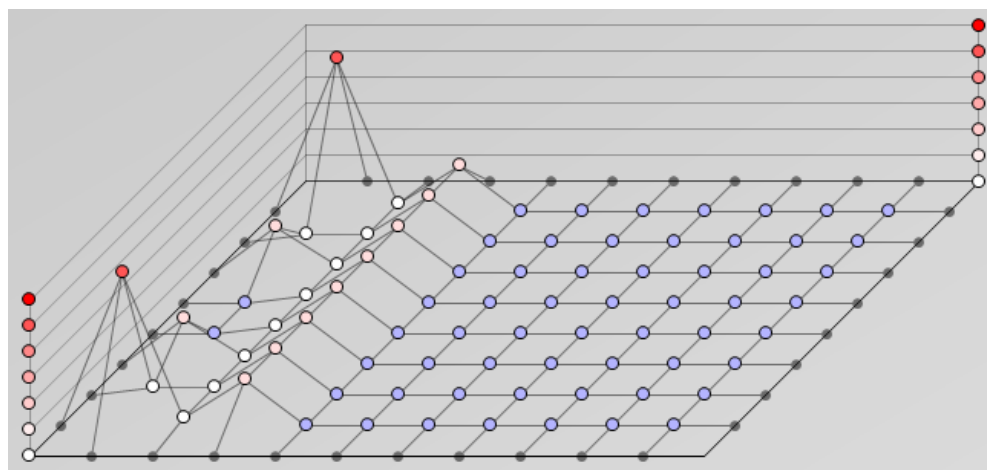
Figure 6-2

```
MeasArray (1,1) = 1  
MeasArray (1,2) = 2  
MeasArray (1,3) = 3  
MeasArray (1,4) = 1  
MeasArray (1,5) = 1  
MeasArray (1,6) = 3  
MeasArray (1,7) = 2  
MeasArray (1,8) = 1
```

As can be seen in the above figure, the displayed measured value can be assigned to another measuring point. Measuring times "2" and "7" were assigned to measuring point "2". Measuring times "3" and "6" were assigned to measuring point "3".

In Runtime, this assignment would look as follows:

Figure 6-3



The figure shows that measuring times "3" and "6" of measuring point "1" have the same value as measuring point "3". However, measuring times "2" and "7" of

### 6.3 Adapting the VBScripts to the purpose

measuring point “1” have the value of measuring point “2”. This assignment corresponds to the program code shown in figure 6-2.

Changing this assignment can, for example, make sense if you want to display setpoint values in the 3D-Grid and not actual values (measured values).

# 7 Operation of the Sample Project of the Application

## 7.1 Overview

The aim of the sample project for this application is to show you how the application is used and how it is used in practice. For this purpose, the project includes two process screens. "Classic.pdl" uses the "Classic" script with the default configuration. The "Anwendungsbeispiel.pdl" (application example) process screen uses the two other scripts that are generated when creating a 3D-Grid.

### Note

Before you can use the project, you have to unzip it and then adjust the computer name in the WinCC Explorer.

## 7.2 "Classic" process screen

As already mentioned, the general script is used here, which is best suited if you want to make significant changes to the functionality of the 3D-Grid.

If you want to "set" the 3D-Grid, select the "Classic" button and then single-click on the "Set 3D-Grid" button. If you have not changed the tag values using the sliders, all measuring points of the 3D-Grid will be colored blue. The color of the scale points on the edge of the grid will also be set.

The "Classic" script works in such a way that the measured values of the measuring points (width) are applied for all measuring times (depth).

Example:

When you set the value of the first measuring point to "100" and then re"set" the grid, you will see that all measuring times of measuring point "1" have the value "100".

Select the "Set 3D-Grid" button in the "Classic" process screen to call a project module (Classic\_Script) or the "GSC\_Classic" procedure from the Global Script Editor. This procedure contains the script from the "3D\_Grid\_CLASSIC\_\*.txt" file.

### 7.3 “Application example” process screen

The aim of the “Anwendungsbeispiel” (application example) process screen is to show a practical configuration for the 3D-Grid. The measured values are acquired at hourly intervals over a period of eight hours (e.g., duration of a production shift).

**Note** To facilitate testing the functionality, the trigger was set to one minute and not to one hour. The data is thus collected every minute.

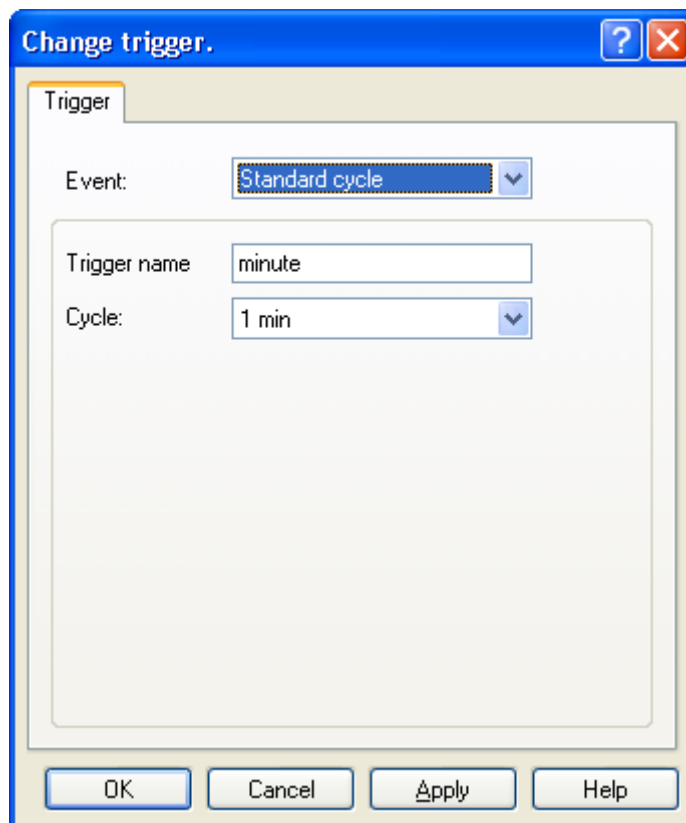
To save the values, internal tags of the previously created “3D-Grid” structure were used.

The “3D-Grid” tag group includes eight x ten tags to save values (eight measuring times, each with ten measuring points) and the “idx\_grid” tag that contains the current measuring time (1-8).

The “GSC\_Anwendungsbeispiel” (application example) Global Script action is used to acquire values every minute. It contains the 3D\_Grid\_GSC\_Action\_\*.txt script.

Acquiring values every minute is implemented with a cyclic trigger (standard cycle: 1 minute).

Figure 7-1



Using the “Set 3D-Grid” button, the grid can be updated at any time. How the display changes depends on the values in the tags.

## 8 References

### 8.1 References

This list is by no means complete and only presents a selection of related references.

Table 8-1

	Topic	Title
/1/	STEP7	Automating with STEP7 in STL and SCL Hans Berger Publicis Corporate Publishing ISBN 3-89578-113-4

### 8.2 Internet links

This list is by no means complete and only presents a selection of appropriate information.

Table 8-2

	Topic	Title
1	Reference to the document	<a href="http://support.automation.siemens.com/WW/view/en/49492528">http://support.automation.siemens.com/WW/view/en/49492528</a>
2	Siemens I IA/DT Customer Support	<a href="http://support.automation.siemens.com">http://support.automation.siemens.com</a>
3	Siemens I IA/DT Online Support	<a href="#">WinCC – Examples of integrated engineering with STEP 7</a>
4	Application example	<a href="#">Example blocks for WinCC and STEP 7</a>
5	Diagnostics	<a href="#">Evaluating WinCC diagnostics files with proposals for solutions by means of info texts and online entries from the Service &amp; Support pages</a>
6	Reference to scripting in WinCC	<a href="#">WinCC Scripting: VBS, ANSI-C, VBA</a>

## 9 History

Table 9-1

Version	Date	Modification
V1.0	August 2011	First edition