SIEMENS

PCS 7

Libraries
APL Style Guide

Programming Manual

10/2012
A5E03860312-02

Legal information

Warning notice system

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

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E03860312-02
Ⓟ 10/2012 Technical data subject to change

Copyright © Siemens AG 2012.
All rights reserved

Table of contents

1 Preface

Introduction

The "Advanced Process Library" (APL) is the standard library of PCS 7 for implementing automation and process control solutions. The APL contains function blocks and the associated faceplates. The faceplates have a uniform design with regard to the operating and monitoring user interface.

The APL Style Guide provides the information, design specifications and rules that you require to adapt project-specific blocks in conformance with APL. You will receive support in these required adaptations by examples of actions.

After all this work has been completed, the adapted project-specific blocks will behave like APL blocks.

Target group

The "APL Style Guide" is intended for persons who use standard APL blocks and who want to additionally adapt project-specific blocks to the specifications of the APL.

Note

The APL Style Guide is not intended to help you in creating completely new technological libraries.

Limitations and requirements

The APL Style Guide supplements the documentation for creating and programming blocks. The APL Style Guide contains solely information about APL-specific properties and functionalities.
The following competent knowledge of PCS 7 is required to work with the APL Style Guide:

- Creating blocks and faceplates
- Working with the screen window technology
- Dynamizing object properties
- Programming blocks in SCL

Validity

The APL Style Guide is valid as of the following version of PCS 7: PCS 7 V8.0.

- Project-specific faceplates created on the basis of the APL library of PCS 7 V8.0 retain their validity.
- The APL style guide is based on the use of PCS 7 template screens and the objects they contain.

Content

The APL Style Guide supplements the Programming Manual *SIMATIC; PCS 7 Process Control System; Programming Instructions for Blocks*.

The APL Style Guide has the following structure and contents:

- Introduction

  Contains information about the purpose of the Style Guide, the target group and the requirements.
- Basics (Page 9)

  Contains additional information about the use of the Style Guide as well as practice-oriented applications with implementation strategies.
- Specifications for block icons (Page 15)

  Contains rules for structuring and configuring project-specific block icons.
- Specifications for faceplates (Page 17)

  Contains rules for structuring and configuring project-specific faceplates.
- Design specifications of the APL (Page 31)

  Contains general specifications for designing block icons and faceplates in conformance with APL.
- Creating project-specific block icons and faceplates
  Contains the required actions for adapting the project-specific blocks:
  - Adapting AS blocks (Page 42) (creating structures and parameters)
  - Creating block icons (Page 72)
  - Creating faceplates (Page 85)

2 Basics

2.1 Using the APL Style Guide

Advantages of blocks in conformance with APL

Creating a PCS 7 project on the basis of the APL has the following advantages:

- Configuration is made easier by using standardized blocks for motors, valves, closed-loop controllers and monitoring.
- Project-specific blocks will receive a uniform design.
- Project-specific blocks are integrated into the APL regarding function and operation.
- You can create project-specific blocks in conformance with APL.

Using the APL Style Guide

The APL Style Guide is typically used as follows:

A project-specific library with technological and project-specific blocks was maintained in an existing project. The standard blocks of the APL are used as new technological blocks together with the project-specific blocks in a new project.

- The program code and the interfaces of the project-specific blocks should remain unchanged as far as possible.
- Because the associated faceplates do not conform to the design of the APL, you will have to carry out adaptations.

Use the rules for designing APL faceplates and the instructions for configuration to carry out the necessary adaptations. The APL Style Guide will show you how to proceed:

- Using colors
- Arrangement of screen objects
- Connecting the faceplate to the process

Functionality of the AV and EventTs blocks

The functionality of the **AV** and **EventTs** blocks is restricted to the standard APL blocks. This functionality is not available for user blocks.

Loss of project-specific adaptations to APL faceplates and APL blocks after the APL has been updated

- Recommendation:
  Use only renamed copies of the APL blocks for project-specific blocks in a project. This approach prevents the creation of inconsistent versions of the APL when you update the library.
- When you adapt existing APL faceplates and APL block icons, these changes will be overwritten by the new version when you update the APL.

2.2 Application strategies

Starting situation

In general the project-specific blocks differ from the APL blocks as follows:

- Structure of the interfaces of the function blocks
- Representation of information at the block icon
- Design of the faceplates
- Functions in the faceplate

Adaptations

- The program code of the function blocks will remain nearly unchanged.
- You adapt functions and interfaces of the project-specific blocks in conformance with APL.
- You have the following possibilities for creating project-specific faceplates:
  - You use, for example, the faceplate structure of an existing APL faceplate as the basis for new faceplates.
  - You create new project-specific faceplates.

Template screens

The following template screens contain the screen objects and block icons you use for your project-specific type:

- "@PCS7ElementsAPL.PDL"

  Contains screen objects for creating faceplates.
- "@PCS7TypicalsAPL... .PDL"

  Contains block icons and static screen objects for valves or motors for automatic placement and updating of the objects.
- "@TemplatesAPL... .pdl"

  Contains block icons and static screen objects for valves or motors for manual placement and updating of the objects.

Note

Versions of the "@PCS7...APL... .PDL" template screen in PCS 7 V8.0

PCS 7 V8.0 contains the following template screens in several versions:

- Versions of "@PCS7TypicalsAPLV... .PDL"
  - @PCS7TypicalsAPLV7.PDL
  - @PCS7TypicalsAPLV8.PDL
- Versions of "@TemplatesAPLV... .PDL"
  - @TemplatesAPLV7.PDL
  - @TemplatesAPLV8.PDL

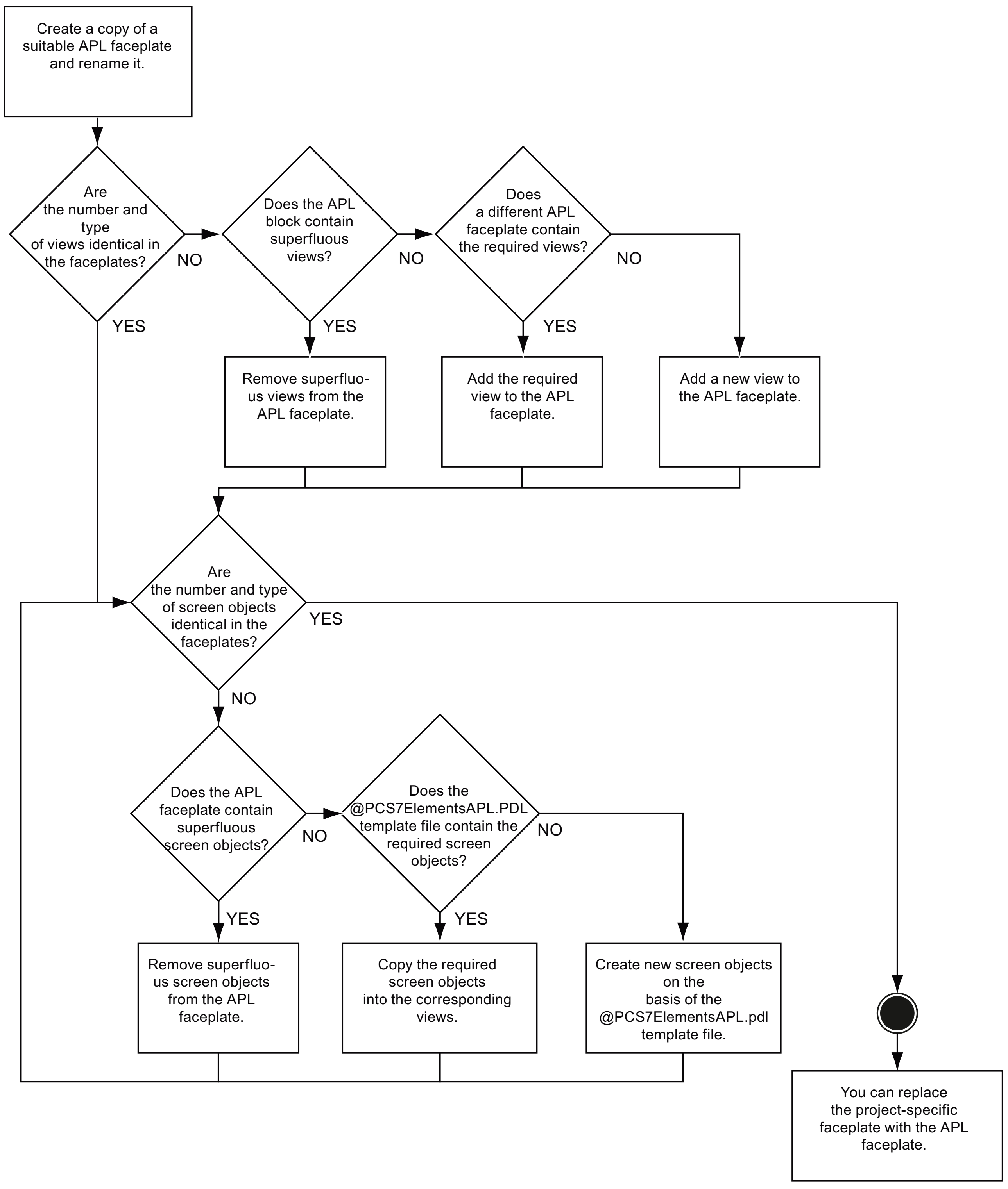
In the manual, the objects of the "@PCS7TypicalsAPLV8.PDL" and "@TemplatesAPLV8.PDL" template screens are used.

For additional information, refer to the chapter "Template screen for faceplates (Page 17)".

Adapting project-specific faceplates

- Check the following points before you create project-specific faceplates:
  - Can I replace a project-specific faceplate directly by an APL block?
  - If there is an APL faceplate whose function is closest to the function of the project-specific faceplate, then proceed as described in the following flowchart.
- If you do not find an APL faceplate whose function is closest to the function of the project-specific faceplate, then create a new faceplate.
  For this purpose, use the screen objects from the @PCS7ElementsAPL.pdl template screen.

Flowchart


Create a copy of a suitable APL faceplate and rename it.
Are the number and type of views identical in the faceplates?
NO
YES
Does the APL block contain superfluous views?
NO
YES
Does a different APL faceplate contain the required views?
NO
YES
Remove superfluous views from the APL faceplate.
Add the required view to the APL faceplate.
Add a new view to the APL faceplate.
Are the number and type of screen objects identical in the faceplates?
YES
NO
Does the APL faceplate contain superfluous screen objects?
NO
YES
Does the @PCS7ElementsAPL.PDL template file contain the required screen objects?
NO
YES
Remove superfluous screen objects from the APL faceplate.
Copy the required screen objects into the corresponding views.
Create new screen objects on the basis of the @PCS7ElementsAPL.pdl template file.
You can replace the project-specific faceplate with the APL faceplate.

Recommendation

Create a project-specific faceplate if possible on the basis of an existing APL faceplate. Copy missing objects from the templates. Only create new screen objects or views if you cannot find a suitable template in the APL.

Specifications for block icons 3

3.1 Template screen for block icons

Description

The "@PCS7TypicalsAPLV8.PDL" and "@TemplatesAPLV8.PDL" template screens contain the preconfigured block icons and static screen objects. By means of drag-and-drop, apply the required block icons and screen objects to your project-specific template screens.

Note

Regular updating of the template screen

Use only block icons from the following template screens for your process pictures:

- "@PCS7TypicalsAPLV8.PDL"
- "@TemplatesAPLV8.PDL"

The up-to-date block icons contained always conform to the correct style and design specifications.

Content

The template screen contains the following objects:

- Static symbols for the status display of motors and valves
- Block icons in different variants for each faceplate of the APL

See also

Adapting block icons (overview) (Page 72)

3.2 Block icon structure

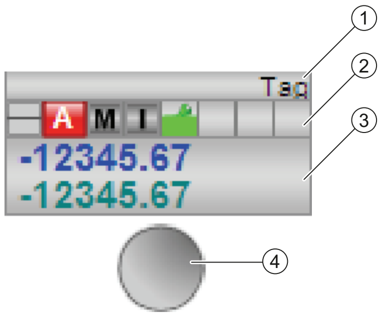
Dimensions

An block icon has the following dimensions:

- Width (without symbol): 101 pixels or 115 pixels
- Width (with symbol): 115 pixels or 151 pixels
- Height: variable

Structure

The following figure shows an example of the structure of a block icon:



①	Display of the instance-specific name (required)
②	Status display for block state (required)
③	Analog value display (optional)
④	Symbol for block operation (optional) You can position the symbol at the top, bottom, right or left.

See also

Adapting block icons (overview) (Page 72)

3.3 Rules for the arrangement of screen objects in block icons

There are no rules for the arrangement of screen objects in block icons.

Recommendation

When you create project-specific block icons, use the block icons from the template screen for block icons as a guideline.

Specifications for faceplates 4

4.1 Template screen for faceplates

Description

The template screen for faceplates contains preconfigured screen objects that you can drag-and-drop into a project-specific faceplate view.

The template screen for faceplates is called "@PCS7ElementsAPL.PDL" and is stored by default in the "GraCS" directory within the PCS 7 OS project.

Note

Regular updating of the template screen

Use only screen objects from the template screen "@PCS7ElementsAPL.PDL" for your project-specific faceplates. This approach ensures that you always use the current screen objects with the correct style and design specifications in your faceplates.

Content

The template screen contains the following objects

- Text boxes and buttons for status displays
- Buttons for calling the binary operating area
- Analog value displays for displaying values from the process
- Buttons for navigating between faceplates
- Text boxes for labeling
- Bar graphs
- Buttons for navigating between faceplate views
- Basic objects (additional information in the WinCC Information System)
- Symbols of the APL (additional information in the APL online help)

See also

Arranging display and operating objects (Page 21)

4.2 Structure of a faceplate view

Dimensions

Each view of a faceplate from the "Advanced Process Library" has the following dimensions:

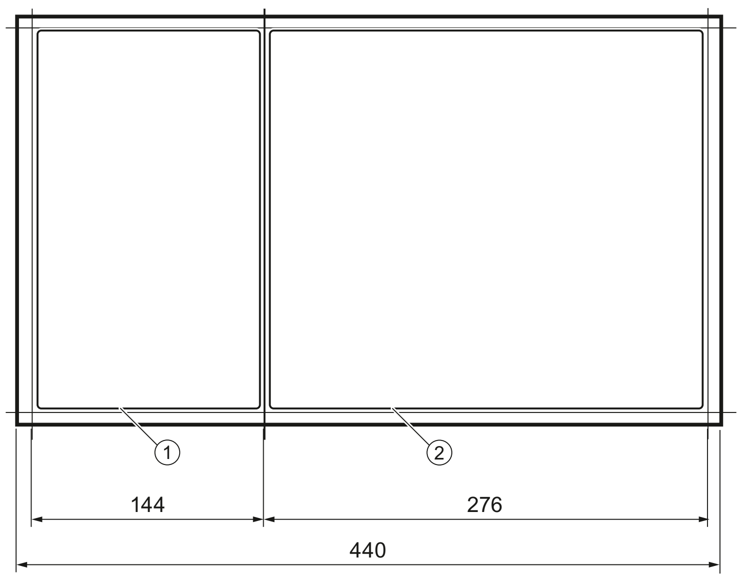
- Width: 440 pixels
- Height: variable
- Margin (top/bottom, right/left): 10 pixels

The specified width does not apply to views that contain a "WinCC AlarmControl" or "WinCC TrendControl". In this case the width depends on the width of the control.

Structure

A faceplate of the "Advanced Process Library" has a two-column design with margin. If you require a completely new view, use the basic design shown below. All views of the APL faceplates use the following basic design for an empty faceplate view.

The following figure shows the grid of an empty faceplate view with the dimensions specified in pixels.



① "Status" area
In this area you monitor the status information of the process.

② "Display and operation" area
In this area you monitor the process values or specify process values using a separate operating area.

Coordinate system and object position

The coordinate system of a screen has its origin in the top left-hand corner. The basis of calculation for the position coordinates of an object is the selection rectangle.

A "static text" as a header in the right-hand column has, for example, the following coordinates:

- Position X: 154
- Position Y: 10

More information on this topic is available under "Working with screens" in the "WinCC Information System".

4.3 Rules for the arrangement of screen objects in faceplates

General specifications for screen objects

As a rule screen objects of the APL are 22 pixels tall. The faceplate views of the APL use the following screen object categories:

- Text box
- Operating and display object
- Dividing line

The following rules apply when you place screen objects in a faceplate view:

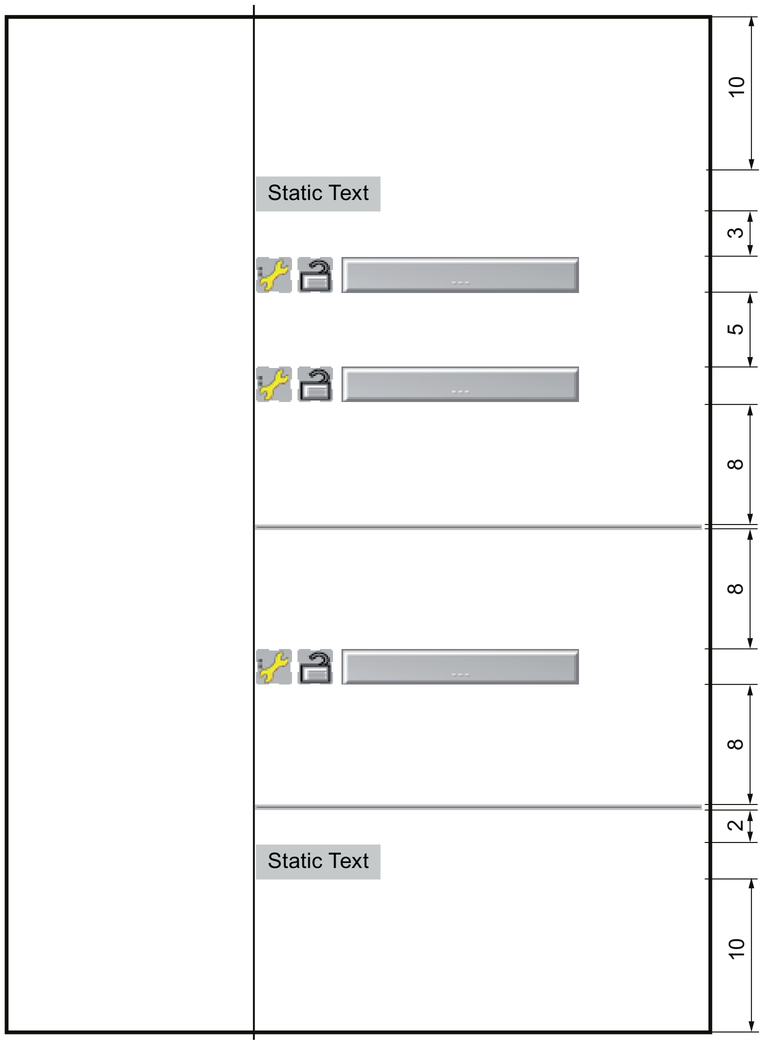
- Place a text box and/or operating and display object at the beginning of the faceplate view.
- Do not place a dividing line at the beginning or end of a faceplate view.
- Observe the distances between the individual screen objects and the margins of the faceplate.

Specifications for top/bottom distances between screen objects

The following table shows the top and bottom distances between the individual screen object types:

Distance between screen objects	Distance in pixels
Dividing line and operating and display object	8
Dividing line and text box	2
Text box and operating and display object	3
Display and operating object and display and operating object	5

The following figure schematically shows the distances displayed in the table:



Rules for right/left distances between screen objects

The distances to the left and right between individual screen objects are determined by the grid of the individual faceplate views.
Generally the distance between two screen objects is 2 pixels.

See also

Structure of the standard view (Page 33)

Structure of the preview (Page 35)

Structure of the batch view (Page 36)

Structure of the limit value view (Page 37)

Structure of the parameter view (Page 38)

Structure of the ramp view (Page 39)

Arranging display and operating objects (Page 21)

4.4 Arranging screen objects in faceplates

4.4.1 Arranging display and operating objects

Introduction

Screen objects from the template screen are often combined to a functional unit in the faceplates, for example, the display of a process value with its permitted limits.

Typical functional units

The APL Style Guide contains a listing of the required screen objects as well as their position in the faceplate view for the following typical functional units:

- Analog value display
- Interlock
- Display and changeover of the operating mode
- Value specification with limit value display
- Bar graph

See also

Configuring the analog value display (Page 21)

Configuring an interlock (Page 24)

Configuring the display and changeover of the operating mode (Page 25)

Configuring the value specification with limit value display (Page 26)

Configuring a bar graph (Page 27)

Rules for the arrangement of screen objects in faceplates (Page 19)

Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)

4.4.2 Configuring the analog value display

Use

Use an analog value display to display or enter analog values. If you also want to display a value graphically, you will have to configure a bar graph. To establish a clear relationship between the analog value display and the bar graph, use the bar color as the frame color of the analog value display.

The following table shows the analog value displays in the template screen and their use:

Object type	Position in the faceplate	Frame color (HTML color code)	Use
APL_ANALOG_OP_DISPLAY	"Display and operation" area	Blue (0000FF)	Display of the setpoint including authorization check
APL_ANALOG_OP_DISPLAY2	"Display and operation" area	Green (00B500)	Display of the following values including authorization check: Representation of large integer values, such as values of counter blocks
APL_ANALOG_OP_DISPLAY3	"Display and operation" area	Dark gray (808080)	Display of the gradient of a ramp including authorization check
APL_TIME_OP_DISPLAY2	"Display and operation" area	Red (FF0000)	Display of a duration
APL_ANALOG_OP_DISPLAY4	"Display and operation" area	Dark gray (808080)	Display of parameter values for a PID controller

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.

Procedure

Information on configuring an analog value display is available in the chapter "APL_ANALOG_OP_DISPLAY analog value display (Page 98)".

See also

Arranging display and operating objects (Page 21)

Structure of a faceplate view (Page 18)

Configuring the highlighting of analog value displays (Page 23)

Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)

4.4.3 Configuring the highlighting of analog value displays

Introduction

You have the following possibilities of highlighting an analog value display in the faceplate view:

- Highlighting by a different color

  When you operate the analog value display, the inscription will be highlighted in color.
- Passivated representation of analog values, for example, limit view of Alarm Limits; when these are deactivated, they will be represented as passivated in the faceplate.

Note

Highlighting in color is also available for the "Checkbox" screen object in addition to the analog value display.

The following table shows the objects for empty analog value displays:

Object type	Position in the faceplate	HTML color code	Use
APL_ANA_EMPTY APL_ANA_EMPTY2 APL_ANA_EMPTY3	"Display and operation" area	Dark gray (808080)	Display of an empty analog value display depending on the value of a block input or output

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.
- Analog value display is configured.

Procedure

Proceed as follows to configure the highlighting of an analog value display:

- To highlight an analog value display in color, select the analog value display.
  In the configuration dialog, set the "Other > ShowSelectionBorder" attribute to "On".
- To display an empty analog value display in addition to an analog value display, place the empty analog value display at the same position as the existing analog value display. Configure the relevant block input or output.

See also

Configuring the analog value display (Page 21)

ShowSelectionBorder (Page 98)

APL_ANALOG_OP_DISPLAY analog value display (Page 98)

4.4.4 Configuring an interlock

Use

Interlocks are process-related deactivations for a faceplate (e.g. motor).

Objects for the interlock

Object type	Position in the faceplate	Use
APL_FACEPLATE3	"Display and operation" area	Buttons for locking and unlocking
APL_LOCK_SYMBOL	"Display and operation" area	Display of the interlock status
APL_QUALITY_CODE2 APL_QUALITY_CODE3	"Display and operation" area	Display of the signal status

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.

Procedure

Proceed as follows to configure an interlock:

1. Copy the required fields from the "@PCS7ElementsAPL.pdl" template screen into your faceplate view.
2. Set the following values to position the objects for the interlock:

Object	Position X	Position Y
Interlock button	284	Any
Symbol for the interlock status	258	"Position Y" of the interlock button
Symbol for the signal status	234	"Position Y" of the interlock button

Result

The interlock is configured.

See also

Arranging display and operating objects (Page 21)

Structure of a faceplate view (Page 18)

4.4.5 Configuring the display and changeover of the operating mode

Use

Displaying and changing the operating mode of aggregates.

The operating mode of a motor or valve is represented in the faceplate by a text box. A separate operating area is provided for changing over the operating mode. Configure an additional button so that the operator can open this operating area.

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.

Procedure

Proceed as follows to configure the display and/or to change the operating mode:

1. Copy the required fields from the "@PCS7ElementsAPL.pdl" template screen to your faceplate view.
   For information about the individual properties of the "Display and operation" area, refer to the section "Binary display with text boxes (Page 116)".
2. Set the following values to position the labeling field:

Object	Position X	Position Y
Labeling field	154	Any
Operating mode display	284	"Position Y" of the labeling field
Button for specifying the operating mode	401	"Position Y" of the labeling field

Result

The operating mode changeover is configured.

See also

Arranging display and operating objects (Page 21)

Structure of a faceplate view (Page 18)

Binary value operation APL_OP_BUTTON (Page 109)

4.4.6 Configuring the value specification with limit value display

Use

To display a permitted range for a process value to the operator, you configure a limit value display in addition to an analog value display. The limit value display consists of two additional fields that you place above and below the analog value display.

Use the "Display range" object for the limit value display. Place the object with the name "Display range" in the "Display and operation" area.

Rules

The following rules apply to value specifications with limit value display:

- Include a maximum of two analog value displays in the limit value display, for example, the "Setpoint" and the "Read back value".
- Observe the following vertical distances between the fields:
  – Distance between limit value display and analog value display: 2 pixels
  – Distance between two analog value displays: 5 pixels

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.

Procedure

Proceed as follows to configure a value specification with limit value display:

1. Copy the required objects from the "@PCS7ElementsAPL.pdl" template screen into your faceplate view.
2. Set the following values to position the objects:

Object	Position X	Position Y
Top limit value display	360	Any
Analog value display	154	["Position Y" of the top limit value display] + 24
Button for specifying the bottom limit value display	360	["Position Y" of the top limit value display] • + 48 (one analog value display) or • + 75 (two analog value displays)

3. Configure the variable which supplies values to the limit value display at the "Other > Value" attribute for every limit value display.
   Result:
   The value specification with limit value display is configured.
4. Assign parameters to the analog value display as described in the section "Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)".

See also

Arranging display and operating objects (Page 21)

Configuring the analog value display (Page 21)

Structure of a faceplate view (Page 18)

4.4.7 Configuring a bar graph

Use

Use a bar graph to display process values graphically. You use a vertical or horizontal bar graph depending on the value you want to display.

The following table shows the bar graphs in the template screen and their use:

Object type	Position in the faceplate	Bar color (HTML color code)	Use
APL_BAR_VERTIC_1	"Status" area	Green (00B500)	Displaying the following values: • Process value • Additional value Typical values: Quantity, temperature
APL_BAR_VERTIC_2	"Status" area	Blue (0000FF)	Display of the setpoint
APL_BAR_GRADIENT_1	"Status" area	Gray (707070)	Display of the gradient of a ramp
APL_BAR_HORIZ_1	"Display and operation" area	Brown (D27A00)	Displaying the manipulated variable
APL_BAR_HORIZ_2	"Display and operation" area	Dark cyan (00CECA)	Displaying the read back value
APL_BAR_TEXT_E	"Status" area	Blue (0000FF)	Displaying the external setpoint ("E")
APL_BAR_TEXT_R	"Status" area	Blue (0000FF)	Displaying the target setpoint of the setpoint ramp ("R")
APL_BAR_LINE_TEXT_E	"Display and operation" area	Brown (D27A00)	Displaying the external manipulated variable ("E")
APL_BAR_LINE_TEXT_R	"Display and operation" area	Brown (D27A00)	Displaying the ramp target value ("R")

Rules

The following rules apply for bar graphs:

- Configure an associated analog value field for each bar graph:
  - Use the bar color for the frame of the analog value field.
  - Place the analog value field either below or next to the bar graph.
- Vertical bar graph:
  - Insert a vertical bar graph in the "Status" area.
  - Position a vertical bar graph centered.
  - Position the objects for displaying the external setpoint and the target setpoint of the setpoint ramp to the left of the vertical bar graph, if necessary.
- Horizontal bar graph:
  - Insert a horizontal bar graph in the "Display and operation" area.
  - Position the objects for displaying the external setpoint and the target setpoint of the setpoint ramp above the horizontal bar graph, if necessary.

Requirement

- Graphics Designer is open.
- "@PCS7ElementsAPL.pdl" is open.
- Faceplate view is open.

Procedure

Proceed as follows to configure a bar graph:

1. Copy the required bar graphs from the "@PCS7ElementsAPL.pdl" template screen into your faceplate view.
2. Make the following settings to align two bar graphs:

Target	Actions
Position vertical bar graphs centered with a distance of 2 pixels.	• Set the values for "Position X": – Left-hand bar graph: 46 – Right-hand bar graph: 75 • Position the displays "E" and "R" to the left of the bar graph, if necessary.
Position the vertical bar graph centered.	• Set the value for "Position X": 61
Position horizontal bar graphs (one or two) left-justified with a distance of 2 pixels.	• Set the value for "Position X": 154 • Set the value for "Position Y" (bottom bar graph): "Position Y" of the top bar graph + 24 pixels • Align the bar graphs vertically. • Position the displays "E" and "R" above the bar graph, if necessary.

3. If you have configured the displays "E" or "R", enter the object name of each corresponding bar graph in the "User-defined > BarGraphName" attribute.
   Result:
   The bar graphs are positioned.
4. Assign parameters to the bar graph as described in the section "Bar graphs (Page 106)".

See also

Arranging display and operating objects (Page 21)

Configuring the analog value display (Page 21)

Configuring the value specification with limit value display (Page 26)

Structure of a faceplate view (Page 18)

Design specifications of the APL 5

5.1 Specifications for operating and display objects

5.1.1 Specifications for labeling

Introduction

All texts or numerical values that are displayed in a faceplate are considered "labeling". The following objects contain text or numerical values:

- Static text
- Button
- Analog value display

Specifications

The following specifications apply to all types of text in faceplates:

Font (FontName)	Arial
Font size (FontSize)	14 point
Bold (FontBold)	No
Italics (FontItalic)	No
Orientation (Orientation)	Horizontal
Y-alignment (AlignmentTop)	Centered

Object-specific specifications

The following object-specific specifications also apply:

Button	
Underline (Underline)	No
X-alignment (AlignmentLeft)	Centered
Static text	
X-alignment (AlignmentLeft)	Left
Analog value display	
X-alignment (AlignmentLeft)	Right
Format (Format)	0.######

5.1.2 Specifications for color

Introduction

The template screens for block icons and faceplates contain the screen objects with the colors currently permitted.

Specifications for color

The following table shows the use of standard colors within the APL:

Color	HTML code	Use
Black	000000	Text
White	FFFFFF	Filling color for "Enabled" state
Gray	808080	Standard color for frame
Red	FF0000	Alarm (high / low), prohibition or lack of authorization
Green	00FF00	Process or supplementary value (e.g. quantity), status "OK" or available authorization
Blue	0000FF	Setpoint, status "within the tolerance"
Yellow	FFFF00	Warning (high / low)
Brown	D27A00	Manipulated variable

If you require additional colors, preferably use the default color range of WinCC.

Specifications for fill patterns

In addition to the colors, fill patterns are defined for the background. The following fill pattern types are permitted:

- Transparent
- Solid
- Pattern

The following table shows the permitted fill patterns for block icons, faceplate views and screen objects:

Fill pattern		Permitted objects
0	▦	Labeling field (static text), analog value display - transparent filling
1	■	Labeling field (static text), analog value display - solid filling
5	▧	Button
6	▥	Vertical bar graph
7	▤	Horizontal bar graph, analog value display, faceplate view, block icon

5.2 Specifications for views

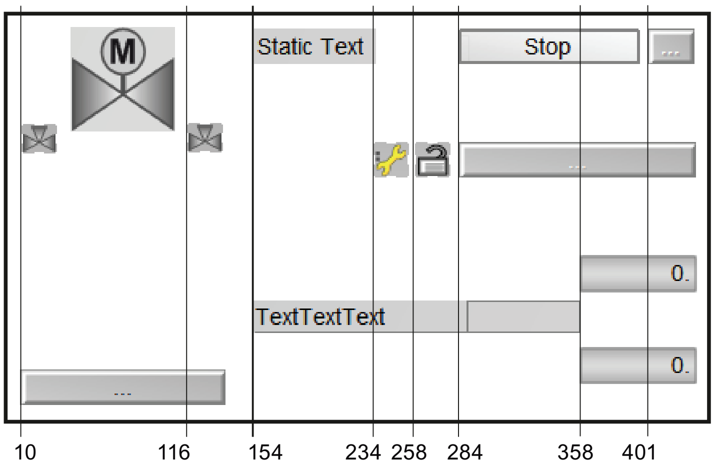
5.2.1 Structure of the standard view

Use

The operator operates and monitors the process in the standard view.

Grid

The following figure shows the typical structure of a standard view including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
10	Start position for symbolic representation of the "Automatic preview" Start position for status boxes and buttons	• The "Status" area is 144 pixels wide. • Status boxes and buttons are 128 pixels wide. • Align the symbolic representation of the "Automatic preview" left-justified to the status boxes. • Position graphical status displays centered to the status boxes in accordance with the following equation: – Position X = ((128/2)+10) - (object width/2) If there are two bar graphs, the object width of the two bar graphs is calculated as follows: – Object width = (2 x width) + 2
116	Start position for symbolic representation of the safety position	• Align the symbolic representation of the "Safety position" right-justified to the status boxes.

"Position X"	Description	Rules
154	Start position of labelings and labeled analog value displays	-
234	Start positions for interlock symbols	• Configure a maximum of two interlock symbols with a width of 22 pixels.
258	Start positions for interlock symbols	• Position the interlock symbols from right to left: – First interlock symbol: Position X = 258 – Second interlock symbol: Position X = 234
284	Start position of buttons	-
358	Start position of fields with upper and lower limits	-
401	Start position of buttons for calling up the operating area	-

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

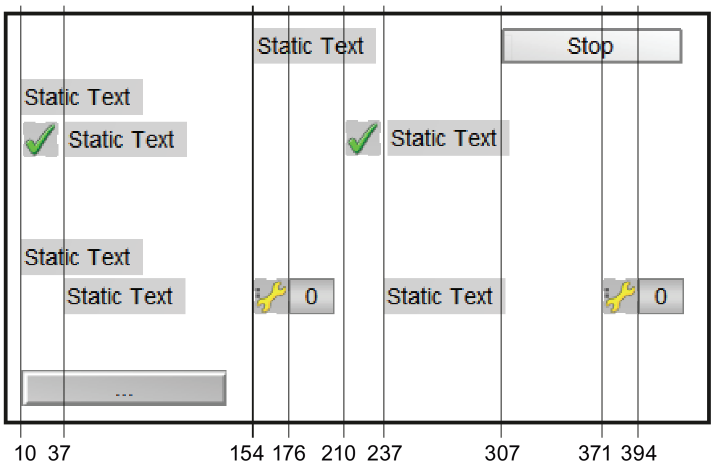
5.2.2 Structure of the preview

Use

The preview shows the parameters that the OS operator can operate in the entire block.

Grid

The following figure shows the typical structure of a preview including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
10	Start position of headers	-
10/210	Start position of status displays (left/right column)	-
37/237	Start position of status labeling (left/right column)	-
154	Start position of labelings and labeled analog value displays	-
154/371	Start position of signal status displays (left/right column)	-
176/394	Start position of control signals (left/right column)	-
307	Start position of buttons	Use buttons with a width of 113 pixels.

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

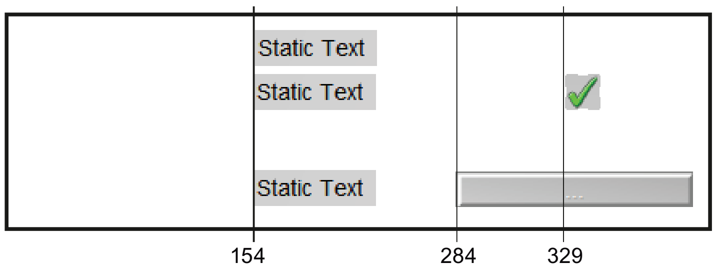
5.2.3 Structure of the batch view

Use

The batch view shows the status of the batch processing.

Grid

The following figure shows the typical structure of a batch view including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
154	Start position of labelings	-
284	Start position of buttons	-
329	Start position of status displays	-

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

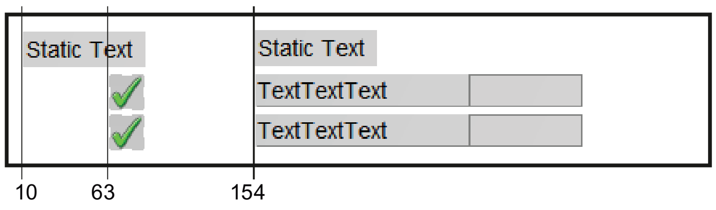
5.2.4 Structure of the limit value view

Use

The operator specifies the limit values in the limit value view.

Grid

The following figure shows the typical structure of a limit value view including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
10	Start position of headers	-
63	Start position of status displays	-
154	Start position of headers and labeled analog value displays	-

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

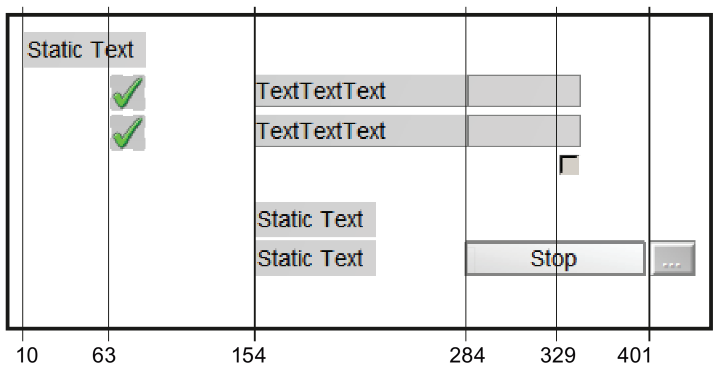
5.2.5 Structure of the parameter view

Use

In the parameter view the operator modifies the parameters (for example, for closed-loop controllers).

Grid

The following figure shows the typical structure of a parameter view including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
10	Start position of headers	-
63	Start position of status symbols	-
154	Start position of labeled analog value displays and labelings	
284	Start position of buttons	Use buttons with a width of 118 pixels.
329	Start position of check boxes	
401	Start position of buttons for calling up the operating area	

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

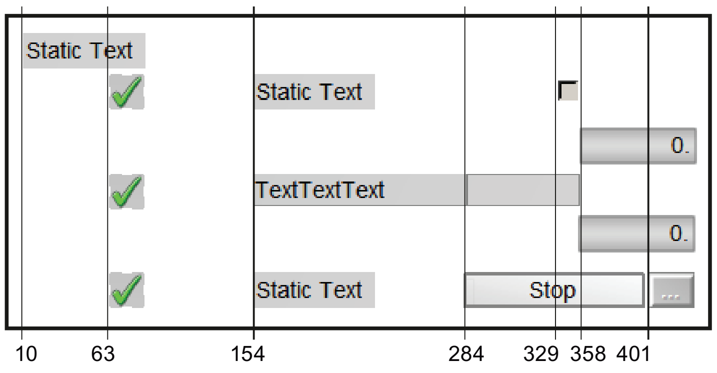
5.2.6 Structure of the ramp view

Use

You control the ramp function in the ramp view.

Grid

The following figure shows the typical structure of a ramp view including the X-coordinates of the grid:



The following table contains the X-coordinates in pixels for the arrangement of objects within the grid:

"Position X"	Description	Rules
10	Start position of headers	-
63	Start position of status symbols	-
154	Start position of labelings and labeled analog value displays	-
284	Start position of buttons	Use buttons with a width of 118 pixels.
329	Start position of check boxes	-
358	Start position of fields with upper and lower limits	-
401	Start position of buttons for calling up the operating area	-

See also

Rules for the arrangement of screen objects in faceplates (Page 19)

6 Creating project-specific blocks in conformance with APL

6.1 Introduction

The following section provides information on creating you own blocks in conformance with APL.

Requirement

- You make changes to the AS block:
  - Adapting inputs and outputs of the AS blocks in conformance with APL
  - Adapting the block code for APL functions
- Use the following as a source for the block icons:

  Standard block icons of the APL
- Use the following as a source for the faceplates:

  Standard faceplates of the APL

Functionality of the AV and EventTs blocks

The functionality of the AV and EventTs blocks is restricted to the standard APL blocks. This functionality is not available for user blocks.

Loss of project-specific adaptations to APL faceplates and APL blocks after the APL has been updated

- Recommendation:

  Use only renamed copies of the APL blocks for project-specific blocks in a project. This approach prevents the creation of inconsistent versions of the APL when you update the library.
- When you adapt existing APL faceplates and APL block icons, these changes will be overwritten by the new version when you update the APL.

6.2 Adapting AS blocks

6.2.1 Adapting AS blocks (overview)

Creating structures and parameters

Function	Section
Creating parameters as structures	Parameters as structures (Page 43)
Displaying status information	Status word (Page 48)
Creating text in the memo view	Memo view (Page 51)
Determining the signal status of an object	Display of the worst signal status (Page 52)
Displaying the units	Display of the measurement unit (Page 54)
Operations using parameters disabled or enabled for specific instances	Parameter OS_Perm / OS1Perm (Page 55)
Programming the behavior patterns of a block	Feature parameter (Page 58)
Designing the visibility of analog auxiliary values depending on their use	Analog value structure (Page 59)
Creating an interface for SIMATIC BATCH in conformance with APL	Connecting SIMATIC BATCH (Page 61)
Using a simulation at APL blocks	Simulation (Page 61)
Creating an "Out of service" function	"Out of service" function (Page 64)
Creating a jump to other faceplates	Jump to other faceplates (Page 67)
Creating the release for maintenance	Release for maintenance (Page 70)

6.2.2 Structures

6.2.2.1 Parameters as structures

APL blocks transfer individual parameters and structured parameters.

Different structure of the parameters

The following parameter types exist for blocks in the APL:

- Parameters that are an interconnection are implemented as a structure.

  Note

  Exception

  The following parameters are interconnected, but are not implemented as a structure.

  - `PV_OutUnit`
  - `PV_Unit`
  - `EN`
  - `ENO`
  - `ANY`

- Parameters that are assigned are not a structure.

  Note

  Exception

  The following parameters are implemented as a structure (without Signal Status):

  - `Feature`
  - `OS_Perm/OS1Perm`

6.2.2.2 Structure and assignment of a structure

The following structures are mainly used in the APL:

- Structure of analog value (see the following example)
- Structure of binary value (same as analog value, the structure element `Value` is of the type BOOL)

Syntax in SCL

The following example shows the declaration of the input parameters of the structure "Analog value":

```
// Structure analog value
     PV:    STRUCT
              Value      : REAL := 0.0;       // Value
              ST         : BYTE := 16#80;     // Signal Status
     END_STRUCT
```

The following example shows the assignment of a structure:

```
// Assignment
     PV.Value    := X1 + X2;
     PV.ST       := 16#78;
```

6.2.2.3 Use of a user-defined data type (UDT)

If you use structured parameters, create these as user-defined data types (UDTs).

Procedure

Create data types in an SCL file as shown in the following example script. Define the symbolic name of the data type as follows:

- "`Type AnaVal`" for an analog value structure

  or
- "`Type DigVal`" for a binary value structure

```
// Example script "UDT51": Creating data types
TYPE AnaVal
        STRUCT
              Value      : Real := 0.0;       // Value
              ST         : Byte := 16#80;     // Signal Status
        END_STRUCT
END_TYPE
```

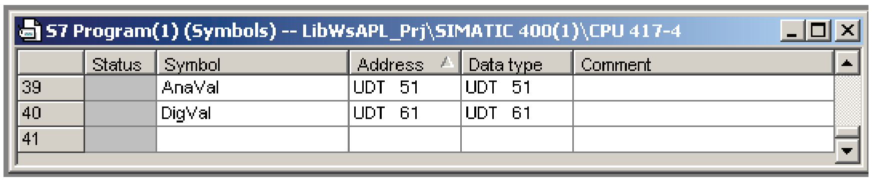
```
// Example script "UDT61": Creating data types
TYPE DigVal
        STRUCT
             Value      : Bool := false;     // Value
             ST         : Byte := 16#80;     // Signal Status
        END_STRUCT
END_TYPE
```

Before you compile the SCL source, assign a UDT number to the symbolic names in the symbol table.

Note

Reserved UDT numbers

The UDT numbers from 1 to 50 are reserved for the data types of the APL.

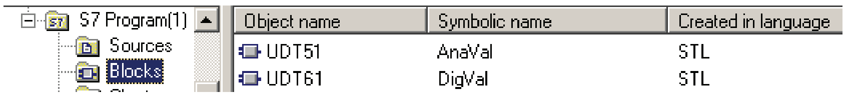


S7 Program(1) (Symbols) -- LibWsAPL_Prj\SIMATIC 400(1)\CPU 417-4

	Status	Symbol	Address	Data type	Comment
39		AnaVal	UDT 51	UDT 51	
40		DigVal	UDT 61	UDT 61	
41					

Figure 6-1 Example: AnaVal = UDT 51 und DigVal = UDT 61

When compiling an SCL source file the user-defined data type is created and stored in the block folder:



Object name	Symbolic name	Created in language
UDT51	AnaVal	STL
UDT61	DigVal	STL

You can now create structure parameters as follows in your block:

```
// Example script: Parameter as a structure
     VAR_INPUT
          PV          : AnaVal;
          FbkOpen     : DigVal;
```

The symbolic name (for example DigVal) will be replaced by the user-defined data type during compilation.

Note

Supply library

You only need the user-defined data types in the block folder to compile the blocks. These user-defined data types do not have to be available in the supply library.

6.2.2.4 Attributes for elements in structures

The APL uses attributes to create individual elements of structures as variables when the OS is compiled.

The attributes start with the prefix "S7_". If the attribute refers to an element of a structure, the prefix is followed by an "x", e.g. "S7_x".

The syntax of these structures remains unchanged.

Rules

Note

Checking the syntax

The SCL compiler does not carry out a syntax check for the content between the quotation marks of an S7_x attribute in the attribute branch.

Ensure that the syntax is observed and that the listing of the element names agrees in an "S7_x" attribute.

The following rules apply to attributes:

- Attributes with prefix "S7_" (without x)

  These attributes apply to the entire structure.

  Example: `S7_visible, S7_dynamik, S7_contact.`

  The syntax is the same as for attributes of unstructured parameters.

- Attributes with prefix "S7_x"

  These attributes always refer to elements within a structure.

  The syntax is a listing of the structure elements assigned to an attribute that are separated by a semicolon.

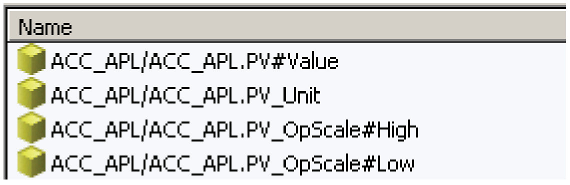
  Example: See example script "Struct_4" ⇒ `PV_OpScale`

  – `S7_m_c := 'true';`

    The expression specifies that you can create variables in the PCS 7 OS for this structure.

  – `S7_xm_c := 'High,true;Low,true;';`

    The expression specifies that the "High" and "Low" elements of the `PV_OpScale` structure are created as variables for the PCS 7 OS.



Name
ACC_APL/ACC_APL.PV#Value
ACC_APL/ACC_APL.PV_Unit
ACC_APL/ACC_APL.PV_OpScale#High
ACC_APL/ACC_APL.PV_OpScale#Low

Figure 6-2 Struct_5

Examples

```
// Example script "Struct_4":
// Create elements of structures as variables
PV          {S7_dynamic := 'true';
               S7_visible      := 'true';
               S7_m_c          := 'true';
               S7_xm_c         := 'Value,true;';
               S7_xshortcut    := 'Value,true;'}
            : STRUCT
               Value           : REAL := 0.0;        // Value
               ST              : BYTE := 16#80;      // Signal Status
            END_STRUCT;

PV_OpScale {S7_m_c:='true';
            S7_xm_c            := 'High,true;Low,true;';
            S7_edit            := 'para';
            S7_xedit           := 'High,para;Low,para;'}
            : STRUCT
               High            : REAL := 100.0;     // High Value
               Low             : REAL := 0.0;       // Low Value
            END_STRUCT ;            // PV - Bar Display Limits for OS
```

The following expression defines the measuring range of the curve control:

- `S7_xtrend:='Value,PV_OpScale.Low,PV_OpScale.High;`

The following expression specifies that an archive variable for the short-term archive is created when an instance is created:

- `S7_xarchive:='Value,shortterm;';`

The following S7_x attributes are available for structures

Attribute	Requirement / property
S7_xm_c	Compulsory with S7_m_c for the structure itself
S7_xshortcut	
S7_xstring_0	
S7_xstring_1	
S7_xedit	
S7_xqc	
S7_xarchive	(is not used by the APL blocks)
S7_xtrend	
S7_contact	
S7_visible	
S7_dynamic	

6.2.3 Status word

Status words (double words) are used for the display of status information in the APL faceplates and APL block icons.

You add these status words to your AS block to use the OS objects.

Recommendation:

Use the SCL command "AT" to allocate the binary information into the double word. With the "AT" command you can create a different view on data types or on the memory area of data types in the data block.

Example:

In the following example "Status1" has been created as the output parameter of the DWORD type:

```
VAR_OUTPUT
            Status1 : DWORD;
            Status1Bits AT Status1 : ARRAY[0..31] OF Bool;
```

- "`Status1Bits`" is a view on the memory area of the double word "`Status1`" in the form of an Array of Bool.
- Use "`Status1Bits`" to address individual bits of this double word, for example, with the following syntax: `Status1Bits[0]:= True;`

  This instruction sets Bit 24 in the double word to TRUE. The assignment of the corresponding bits is shown in the following section.

Actions during loading

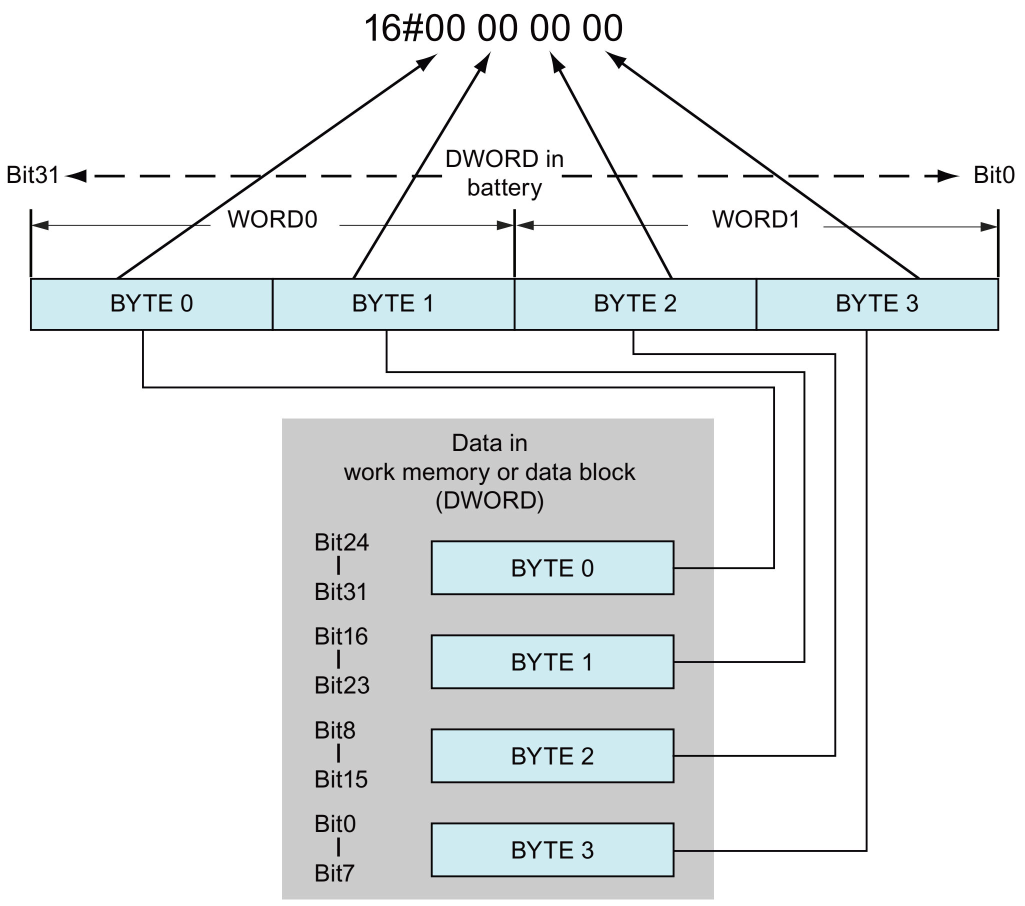
When a double word is loaded from the memory into the accumulator, the bytes will be swapped as follows:

- The High WORD and Low WORD are swapped in the double word.
- The High Byte and Low Byte are swapped in the word.

Result in the accumulator:

- The byte with the highest address area in the DB is Bits 0 to 7 of a double word.
- The byte with the lowest address area in the DB is Bits 24 to 31 of a double word.

The "AT" command always specifies its view in ascending order using the data block or memory address area.



We recommend that you use a user-defined data type (UDT) for the processing of individual bits in a double word.

The following example shows the structure of the user-defined data type:

```
// Example script: Data type for allocating the bits in double words
TYPE SBit32inDWORD
    STRUCT
        // highest Byte in accu                BYTE0 in memory
            X24 : BOOL;                        // 0.0
            X25 : BOOL;
            X26 : BOOL;
            X27 : BOOL;
            X28 : BOOL;
            X29 : BOOL;
            X30 : BOOL;
            X31 : BOOL;                        // 0.7
        // second highest Byte in accu         BYTE1 in memory
            X16 : BOOL;                        // 1.0
            X17 : BOOL;
            X18 : BOOL;
            X19 : BOOL;
            X20 : BOOL;
            X21 : BOOL;
            X22 : BOOL;
            X23 : BOOL;                        // 1.7
        // third highest Byte in accu          BYTE2 in memory
            X8  : BOOL;                        // 2.0
            X9  : BOOL;
            X10 : BOOL;
            X11 : BOOL;
            X12 : BOOL;
            X13 : BOOL;
            X14 : BOOL;
            X15 : BOOL;                        // 2.7
        // lowest Byte in accu                 BYTE0 in memory
            X0  : BOOL;                        // 3.0
            X1  : BOOL;
            X2  : BOOL;
            X3  : BOOL;
            X4  : BOOL;
            X5  : BOOL;
            X6  : BOOL;
            X7  : BOOL;                        // 3.7
    END_STRUCT
END_TYPE
```

Addressing a bit in a double word

The following example shows the addressing of Bit0 in the double word:

```
// Example script: Addressing a bit in a double word
VAR_OUTPUT
    Status1                   : DWORD;
    Status1Bits AT Status1    : SBits32inDWORD;
END_VAR
Begin
    Status1Bits.X0            := True;
```

6.2.4 Memo view

The "AUTHOR" block attribute of the corresponding AS block has to start with the characters "AdvLib". When the OS is compiled, OS-internal variables will be created for the memo view.

The variables required for memo view are:

- "`xxx.#TextPermanent`" variable
- "`xxx.#StatusPermanent`" variable

Additional information about the memo view is available in the online help of the Advanced Process Library.

6.2.5 Display of the worst signal status

The FC 369 function (symbolic name `SelST16`) determines the worst signal status in the APL blocks. This status will be displayed in the following locations:

- In the block icon
- In the faceplate overview

Procedure

Create the variable ST_Worst as an output parameter in the AS block, as shown in the example script "ST_Worst".

You can accept the object in the faceplate overview and in the block icon without a change.

```
// Example script "ST_Worst": Creating a variable
ST_Worst     {S7_visible:'false';s7_m_c:='true'}
                  :BYTE := 16#80     ; // Worst Signal Status
```

Elements of the function

The table below shows the input parameters of the SelST16 function:

Parameter	Description
`InST`	Structure with 16 elements of the type BYTE (b0 to b15) You can apply up to 16 signal statuses at this structure.
`Num`	Number of used elements of `InST` (integer)
`SelPrio`	Priority scheme, best or worst signal status (Integer) For information, refer to the online help of the APL.

Return value of the function

Parameter	Description
RetVal	Output of the best or worst Signal Status of the input structure `InST`

Script

```
// Example script: Declaration of the temporary variables
TempInST: STRUCT
b0 : BYTE;
b1 : BYTE;
b2 : BYTE;
b3 : BYTE;
b4 : BYTE;
b5 : BYTE;
b6 : BYTE;
b7 : BYTE;
b8 : BYTE;
b9 : BYTE;
b10 : BYTE;
b11 : BYTE;
b12 : BYTE;
b13 : BYTE;
b14 : BYTE;
b15 : BYTE;
END_STRUCT;
ST_Worst BYTE;
```

```
// Example script: Program call of the function
TempInST.b0 := X1;
TempInST.b1 := X2;
TempInST.b2 := X3;
ST_Worst := SelSt16(InST := TempInST, Num := 3, SelPrio := 0);
```

6.2.6 Display of the measurement unit

APL has its own parameter of the type `INTEGER` for displaying measurement units.

Procedure

Create a parameter for each measurement unit you want to display.

- If you configure a value conforming to the standard IEC 61158 at the parameter, the corresponding unit will be displayed. A corresponding table is available in the online help of the APL.
- If the value amounts to 0, the text entered in the "`S7_unit`" attribute will be displayed.

```
// Example script: Creating a parameter for the measurement unit
PV_Unit {
            S7_m_c:='true';
            S7_unit:='l/sec'
            } : INT; //Unit
```

6.2.7 Parameter OS_Perm / OS1Perm

Use the input parameters "OS_Perm" or "OS1Perm" to disable or enable individual operations for specific instances through parameter assignment.

Note

If "OS_Perm" is not sufficient, the parameter "OS1Perm" is used at the APL blocks.

Creating parameters

In the case of OS-operable blocks it is advisable to create the "OS_Perm" or "OS1Perm" parameter as a structure, as shown in the example script "OS_Perm" .

```
// Example script OS_Perm:  parameter "OS_Perm"  as a structure
OS_Perm    {S7_visible:='false'}
        :STRUCT
            Bit0: BOOL:=1;            //1 = Operator can enable
                                      accumulation
            Bit1: BOOL:=1;            //1 = Operator can switch off
                                      accumulation
             "
             "
             "
            Bit31: BOOL:=0;           // Not used
        END_STRUCT;                   // Operator permissions
        ArrOS_Perm AT OS_Perm :
ARRAY[0..3] of BYTE;
```

Compiling the OS

The "OS_PermOut" and "OS_PermLog" parameters have to be created as outputs. When the OS is compiled, these parameters will be created as variables:

- "OS_PermOut" variable (output double word)

  The "OS_PermOut" output indicates whether an operation is generally enabled at the block.

```
// Example script: "OS_PermOut" variable
OS_PermOut    {S7_visible:='false';S7_m_c:='true'}
        :DWORD:=16#FFFFFFFF; //Parameterized Permissions
ArrOS_PermOut AT OS_PermOut : ARRAY[0..3] of BYTE;
```

- "OS_PermLog" variable (output double word)

  The "OS_PermLog" output indicates whether an operation is enabled generally at the block and process-conditionally.

  In the faceplate, "OS_PermLog" will be used as the operator control enable for the used operating objects.

```
// Example script: "OS_PermLog" variable
OS_PermLog    {S7_visible:='false';S7_m_c:='true'}
        :DWORD:=16#FFFFFFFF; //Permissions with Process conditions
ArrOS_PermLog AT OS_PermLog : ARRAY[0..3] of BYTE;
```

View of structures

The "AT" command provides a view of the structures:

Structure	View	Form	Example
OS_Perm	ArrOS_Perm	ARRAYof BYTE	See example script OS_Perm
OS_PermOut	ArrOS_PermOut	ARRAYof BYTE	See example script OS_PermOut
OS_PermLog	ArrOS_PermLog	ARRAYof BYTE	See example script OS_PermLog

With these views you can transfer the "OS_Perm" input structure to the output double word.

```
// Example script "OS_PermLog":
Transferring the input structure to the output double word
//*********************************************************************
// Copy Data from Structure OS_Perm to
//   DWORD OS_PermOut by means of AT command
//*********************************************************************
ArrOS_PermOut[3]:=ArrOS_Perm[0];
ArrOS_PermOut[2]:=ArrOS_Perm[1];
ArrOS_PermOut[1]:=ArrOS_Perm[2];
ArrOS_PermOut[0]:=ArrOS_Perm[3];
```

Note

If you want to monitor a double word with the SCL command "AT" in the form of bit, byte or word, observe the following points:

- The High Byte and Low Byte are swapped in the word.
- The High WORD and Low WORD are swapped in the double word.

To ensure that the bits of the structure in the output double word are stored correctly, they have to be assigned as shown in the example script "OS_PermLog".

Creating process-related enables

Process-related enables are required for the "OS_PermLog" output.

We recommend that you create temporary variables and views of these variables as shown in the example script "OS_Perm3".

```
// Example script "OS_Perm3": Temporary variables for OS_PermLog
// temporary Parameters for OS_PermLog
tmpDW_OS_Perm:DWORD;
arrtmpDW_OS_Perm AT tmpDW_OS_Perm: ARRAY[0..3] of BYTE;
sttmpDW_OS_Perm AT tmpDW_OS_Perm: STRUCT;
                                        Bit0: BOOL;
                                        Bit1: BOOL;
                                          "
                                          "
                                          "
                                        Bit31: BOOL;
                                    END_STRUCT;
```

Creating a parameterized enable with the process-related enable

The example script "OS_Perm42" shows how to combine the parameterized enable with the process-related enable.

The corresponding bits are written directly to the temporary double word "tmpDW_OS_Perm" through the use of the "AT" command. After the corresponding bytes have been swapped in the temporary double word, writing will be carried out to the "OS_PermLog" output double word.

```
// Example script "OS_Perm42": Connecting a parameterized enable with the
process-related enable
//*******************************************************************
// OS_Perm and process releated permissions
//*******************************************************************
sttmpDW_OS_Perm.Bit0:=OS_Perm.Bit0 AND NOT HiAlmAct;
                                        //permission enable accumulation
sttmpDW_OS_Perm.Bit1:=OS_Perm.Bit1;     //no further process condition
             "
             "
             "
sttmpDW_OS_Perm.Bit31:=OS_Perm.Bit31;   //no further process condition

ArrOS_PermLog[3] := arrtmpDW_OS_Perm[0];
ArrOS_PermLog[2] := arrtmpDW_OS_Perm[1];
ArrOS_PermLog[1] := arrtmpDW_OS_Perm[2];
ArrOS_PermLog[0] := arrtmpDW_OS_Perm[3];
```

6.2.8 Feature parameter

Use the "Feature" parameter to program the behavior patterns of the block.

Procedure

Create the following structure:

```
// Example script: Parameter "Feature"
Feature S7_visible:='false';
        S7_xedit:='Bit0,para;Bit1,para;Bit2,para;Bit3,para;'}
        :STRUCT
            Bit0: BOOL:=0; // common Featurebit not used in this block
            Bit1: BOOL:=0; // common Featurebit not used in this block
            Bit2: BOOL:=0; // common Featurebit not used in this block
                  "
                  "
            Bit5: BOOL:=0;
                // 1 = Alarm Output activ with 0,
                // individual block specific Feature Bit
                  "
                  "
            Bit31: BOOL:=0; // Not used
        END_STRUCT; // Status of various features
```

Note

In the program part take into consideration the bits of the "Feature" parameter at the positions at which you program the different behavior patterns.

6.2.9 Analog value structure

Analog auxiliary values are displayed in the faceplates. For this purpose you use the input parameter of the type "`AnalogValue`" as structure. The visibility depends on whether the input parameter is interconnected.

Displaying analog auxiliary values

The associated AS block Signal Status (must evaluate **16#FF**) and transfer this information in the status word to the OS.

Signal Status is used to detect whether the input parameter is interconnected:

- **Signal Status = 16#FF**

  The input parameter is not interconnected and will not be displayed in the faceplate.

  The Signal Status 16#FF is an internal APL status.
- **Signal Status not equal to 16#FF**

  The input parameter is interconnected and will be displayed in the faceplate.

Creating an input parameter of the "AnalogValue" type as a structure

Create the input parameter as shown in the following example script.

In this example script, the information whether the input parameter is interconnected or not will be transferred through the status word 1 Bit 5.

Note

Signal Status must have 16#FF as the predefined value.

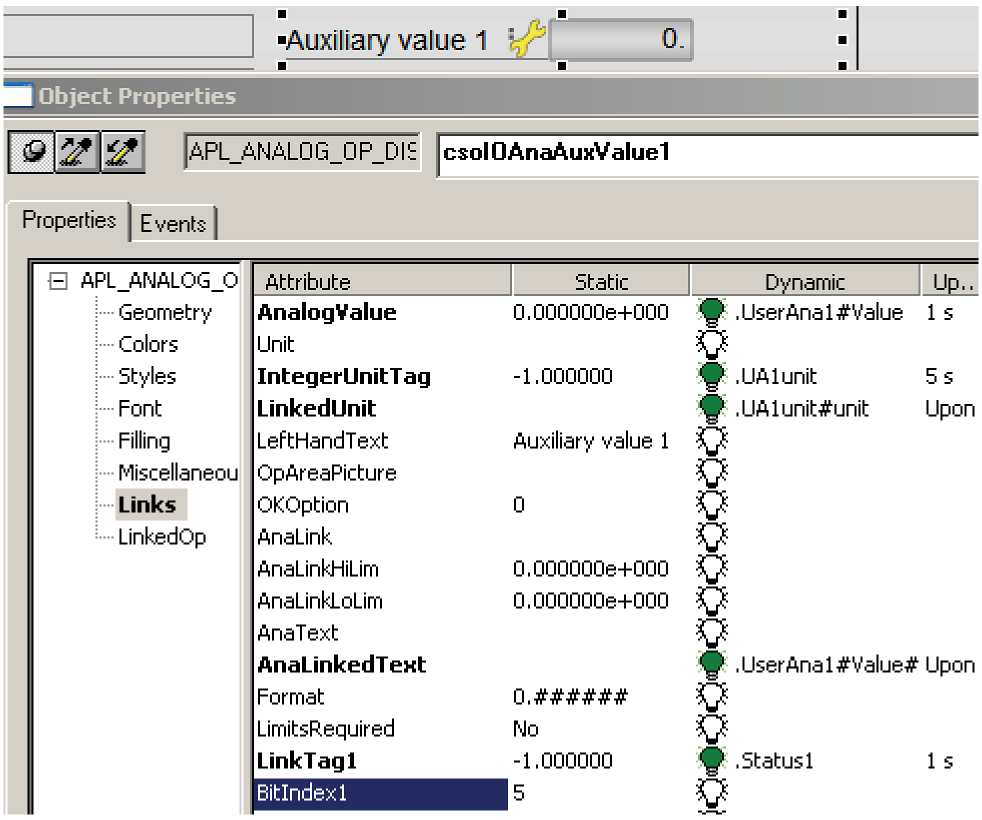
```
// Example script: Creating parameters
VAR_INPUT
   Userana1   {S7_visible      := 'false';
                S7_m_c          := 'true';
                S7_xm_c         := 'Value,true;';
                S7_xqc          := 'Value,true;';
                S7_xshortcut    := 'Value,Auxiliary value1;'}
              : STRUCT
                Value           : REAL := 0.0;      // Value
                ST              : BYTE := 16#FF;    // Signal Status
              END_STRUCT;                           // User Analog Input 1

   UA1unit    {S7_visible      := 'false';
                S7_m_c          := 'true';
                S7_unit         := ''}
              :INT:=0;                              // Unit of UserAna1
END_VAR

BEGIN
     dwOutStatus1.X5:=NOT (UserAna1.ST=16#FF);
```

Copying an object

For displaying auxiliary analog values in the faceplate, copy the following object from the template to your faceplate view:

APL_ANALOG_OP_DISPLAY (@PCS7ElementsAPL.pdl; Display analog Value; Setpoint)

- Set the AnalogValue attribute to .UserAna1#Value..
- Adapt the following attributes in the "Links" property as shown in the example picture.
  - Delete the connections to attributes not shown in the example picture.
  - Set the status word to "LinkTag1" and configure the appropriate bit at "BitIndex1" . The script linked with the "LinkTag1" property controls the visibility of the display via the "If changed" event.
- Set the "LineColor" attribute in the "Color" property to black for displaying the frame.



Attribute	Static	Dynamic	Up..
AnalogValue	0.000000e+000	.UserAna1#Value	1 s
Unit			
IntegerUnitTag	-1.000000	.UA1unit	5 s
LinkedUnit		.UA1unit#unit	Upon
LeftHandText	Auxiliary value 1		
OpAreaPicture			
OKOption	0		
AnaLink			
AnaLinkHiLim	0.000000e+000		
AnaLinkLoLim	0.000000e+000		
AnaText			
AnaLinkedText		.UserAna1#Value#	Upon
Format	0.######		
LimitsRequired	No		
LinkTag1	-1.000000	.Status1	1 s
BitIndex1	5		

6.2.10 Connecting SIMATIC BATCH

Declare the following input parameters for an interface in conformance with APL to SIMATIC BATCH.

```
//SIMATIC BATCH interface
BatchEn {S7_visible := 'false'}           // Enable remote operation of controller by Batch recipe
 :BOOL := false;
BatchID {S7_visible := 'false'; S7_m_c := 'true'}              // Current Batch ID (number)
 :DWORD := 16#00;
BatchName {S7_visible := 'false'; S7_m_c := 'true'}            // Current Batch name
 :STRING [32];
StepNo {S7_visible := 'false'; S7_m_c := 'true'}               // Batch step number
 :DWORD := 16#00;
Occupied {S7_visible := 'false'}                               // Occupied by Batch
 :BOOL := false;
```

6.2.11 Simulation

The following options are available for the simulation in APL blocks.

- Block-external simulation
- Block-internal simulation

For additional information, refer to the APL online help.

Take the following information into account for the block-internal simulation at blocks in conformance with APL.

Simulation of digital process values

The block must have a "SimOn" input parameter for the simulation of digital process values. You operate this input parameter via the faceplate.

When a block with a digital process value input is in simulation (for example, a drive block), the feedbacks will be simulated in accordance with the control.

This is why corresponding feedback outputs are always declared for the feedback inputs.

- When the simulation is active, these feedback outputs will be derived from the control.
- When the simulation is not active, these feedback outputs will be transferred directly to the feedback inputs.

Motor example:

- Feedback input "FbkRun"
- Feedback output "FbkRunOut"

Simulation of analog process values

When a block with an analog process value input is in simulation, you will be able to change this process value from the faceplate. Use the following example script to simulate analog process values:

```
// Example script: Simulation of analog process values
VAR_INPUT
    SimOn   {S7_visible := 'false';
            S7_m_c := 'true'}
            :BOOL := FALSE;            //Simulation On/Off
    SimPV    {S7_visible := 'false';
            S7_m_c:= 'true'}
            :REAL :=0.0;            //Simulation Value
END_VAR
VAR_OUTPUT
    PV_Out    {S7_xqc:='Value,true;';
            S7_dynamic:='true';
            S7_m_c:= 'true';
            S7_xm_c:='Value,true;';
            S7_xshortcut:='Value,PV;';
            S7_xarchive:='Value,shortterm;';
            S7_xtrend:='Value,PV_OpScale.Low,PV_OpScale.High;'}
            :AnaVal; //Process Value (Analog Output)
END_VAR

// Code selection - Simulation
    IF SimOn THEN
        PV_Out.Value    := SimPV;
        PV_Out.ST       := 16#60;
    ELSE
        PV_Out.Value := PV.Value;
        SimPV        := PV_Out.Value;
        PV_Out.ST    := PV.ST;
    END_IF;

    dwOutStatus1.X6:=SimOn;
```

Display and operation

- Copy the following objects from the template to display and operate the "Switch simulation on/off" function in the faceplate. Add these objects to your faceplate view.
  - APL_OP_BUTTON (@PCS7ElementsAPL.pdl; Open binary operation area; Normal)
  - APL_MULTI_TEXT2 (@PCS7ElementsAPL.pdl; Status display text; Text display)

  For additional information refer to the chapter "Binary display with text boxes (Page 116)".
- Copy the following object from the template for the analog value "PV". Add this object to your faceplate view:
  - APL_ANALOG_OP_DISPLAY (@PCS7ElementsAPL.pdl; Display analog value; setpoint)

  For additional information refer to the chapter "Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)".

Note

Ensure here that the corresponding status word / status bit for "SimOn" is interconnected with the operator control enable ("LinkTag3" property) instead of "OS_PermLog".

6.2.12 "Out of service" function

You deactivate all functions of the block with the "Out of operation" operating mode. No incoming or outgoing messages are generated. Only an operating mode switchover is possible.

For additional information, refer to the online help of the APL.

Procedure

Create the interface and the related program section in your block, as shown in the OutOfService_1 example code.
Note the following information about the parameters.

Example script "OutOfService_1"

```
VAR_INPUT
   OnOp     {S7_m_c:='true'; S7_link:='false'; S7_visible:='false'}
            :BOOL := FALSE;       // 1=On  Mode: On Mode by Operator
   OosOp    {S7_m_c:='true'; S7_link:='false'; S7_visible:='false'}
            :BOOL:=FALSE;         // 1=Oos Mode: Oos Mode by Operator
   OosLi    {S7_dynamic:='true'}
            :DigVal;              // 1=Oos Mode: Oos Mode by Field Signal
   OS_Perm  {S7_visible:='false'}
            :STRUCT
               Bit0: BOOL :=1;    // 1 = Operator can switch to On
               Bit1: BOOL :=1;    // 1 = Operator can switch to OOS
            END_Struct;
END_VAR
VAR_OUTPUT
   OosAct   {S7_dynamic:='true';
            S7_m_c:='true';
            S7_xm_c:='Value,true;'}
            :DigVal;                   // Out of service is active
   OnAct    {S7_dynamic:='true'}
            :STRUCT
               Value : BOOL := true;   // Value
               ST    : BYTE := 16#80;  // Signal Status
            END_STRUCT;                // On Mode is active
END_VAR
VAR
   SxOosLi  :BOOL := FALSE;
END_VAR
```

Example script "OutOfService_1"

```
// CODE Selection
   IF OnOp THEN
            OosAct.Value := false;
            OnOp := false;
   ELSIF OosOp OR (OosLi.Value AND Feature.Bit1 AND NOT SxOosLi) THEN
            OosAct.Value := true;
   END_IF;
   OosOp := false;
   SxOosLi := OosLi.Value;
   OnAct.Value := NOT OosAct.Value;
   IF OosAct.Value THEN // OutOfService conditions
            OS_PermLog := (16#1 AND OS_PermOut);
   ELSE                          // Not in mode "Out of service"
   END_IF
```

Parameters for selecting the operating mode

Use the following parameters to select the operating mode of the block:

- Parameter "`OnOp`" switches to the "On" operating mode
- Parameter "`OosOp`" switches to the "Out of operation" operating mode

Note

Blocks with manual / automatic operating mode changeover

A changeover to "Out of operation" is only permitted in manual mode.

Required actions

1. Allocate a bit for the "Out of operation" status in a status word of the AS block.
   The maintenance status is shown in the block icon.
   When copying an APL block icon, you may need to make the following adjustments:
   – The status word
   – The bit directly at the extended status display
2. The maintenance status is shown in the faceplate.
   Adapt the status display for the maintenance status "Out of operation" in the screen @PG_MyAPLFP_Overview, if necessary:
   – The status word
   – Configure the appropriate bit for the extended status display of the overview faceplate.

   Note

   The example script OutOfService_1 shows how the block is to behave.
   - The values will be frozen after the transfer.
   - All operations will be disabled except for switching back to the "On" operating mode.

For additional information refer to the chapter "General specifications in the screen @PG_MyAPLFP_Overview.pdl (Page 89)".

Display and operation

Proceed as described in the following paragraphs to display and operate in the faceplate of the operating mode "On" / "Out of operation":

- Binary value operation
  Section "Binary value operation APL_OP_BUTTON (Page 109)"
- Multitext
  Section "Binary display with text boxes (Page 116)"

The following displays are set to invisible in "Out of operation" mode:

- Group display of the block icon
  The remaining displays are shown with a gray background.
  In the block icon this representation is controlled via the "OosAct" output parameter.
- Group displays of the overview faceplate
  When "Out of service" is enabled, an empty group display is superimposed on the group display with a bit in the status word.
  Use bit 3 in the status word for this purpose or adapt the dynamic dialog of the group display object to Bit 3.

6.2.13 Jump to other faceplates

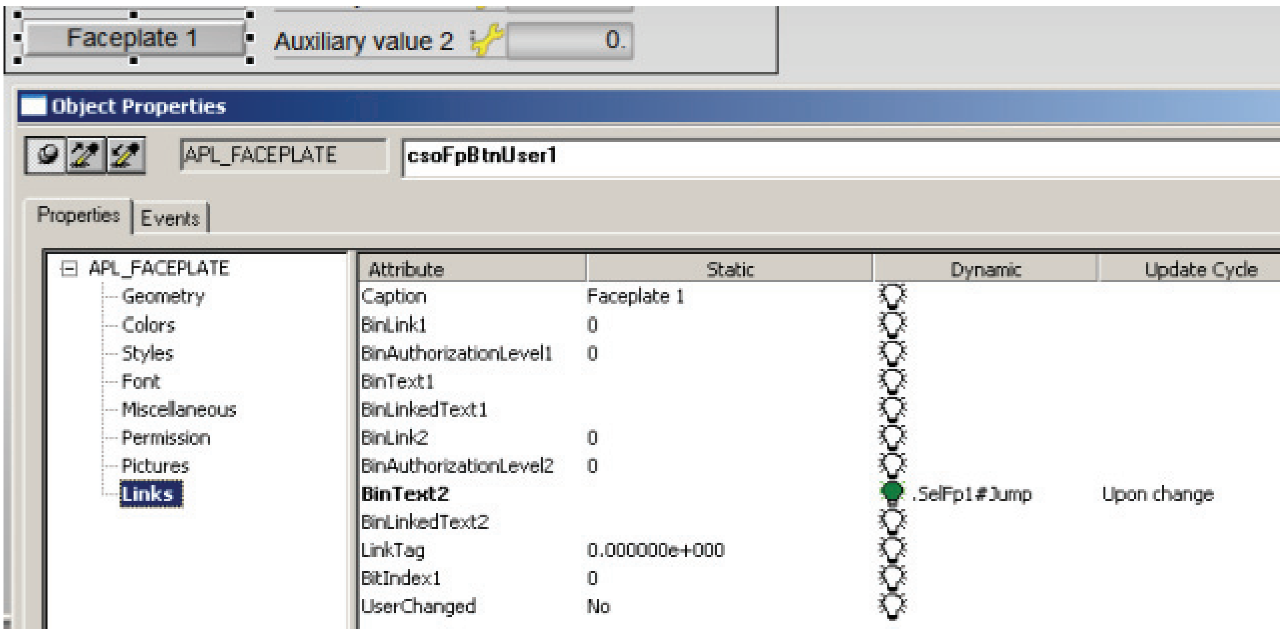
Create jump buttons in the APL faceplates for jumps to other faceplates.

Procedure

Copy the jump buttons and create the parameters "SelFp1" and "SelFp2" .

Jump buttons:

- Jump button 1 is located in the standard view object name "csoFpBtnUser1" (object type "APL_FACEPLATE").
- Jump button 2 is located in the preview object name "csoFpBtnUser2" (object type "APL_FACEPLATE").



APL_FACEPLATE | csoFpBtnUser1

Attribute	Static	Dynamic	Update Cycle
Caption	Faceplate 1		
BinLink1	0		
BinAuthorizationLevel1	0		
BinText1			
BinLinkedText1			
BinLink2	0		
BinAuthorizationLevel2	0		
BinText2		● .SelFp1#Jump	Upon change
BinLinkedText2			
LinkTag	0.000000e+000		
BitIndex1	0		
UserChanged	No		

Figure 6-3 SelFp1_1

Parameter:

- The parameters are of the type ANY and can thus be interconnected to any parameter of another block.
- The parameters have the CFC attribute `BLK_Jump`.

Compiling the OS

- If the `BLK_Jump` attribute is set as in the example script SelFp1_2", an OS-internal variable "xxx.SelFp1#Jump" will be created for the instance during compilation of the OS.
- If the parameter is interconnected, a text reference with the entry of the OS-tagname of the interconnected block will be created.

The reference to the text reference is contained in the "`xxx.SelFp1#Jump`" variables.

```
Example script: Variables with reference to the text reference
SelFp1       {BLK_Jump:='1';S7_visible:='false'}
             :ANY;                     //Select Faceplate1
SelFp2       {BLK_Jump:='1';S7_visible:='false'}
             :ANY;                     //Select Faceplate2
```

SelFp1_2

OS block symbol

Attributes such as `S7_string` or `S7_shortcut` cannot be assigned to an Anypointer . Instance-specific labeling of the buttons is stored in the block icon:

Properties -> UserButtonText1, UserButtonText2

Assignment of a text for labeling of a jump button from the CFC block

Option 1:

```
Example script: Creating a parameter of the type BOOL
SelFp1_3    {BLK_Jump:='1';
            S7_m_c:='true';
            S7_string_1:='Jump to xxx'
            }
            : BOOL; //Select Faceplate 3
```

SelFp1_3

1. Create a parameter of the type BOOL , for example, instead of an Anypointer, and assign a text for labeling of the jump button to the `S7_string1` attribute.

Note

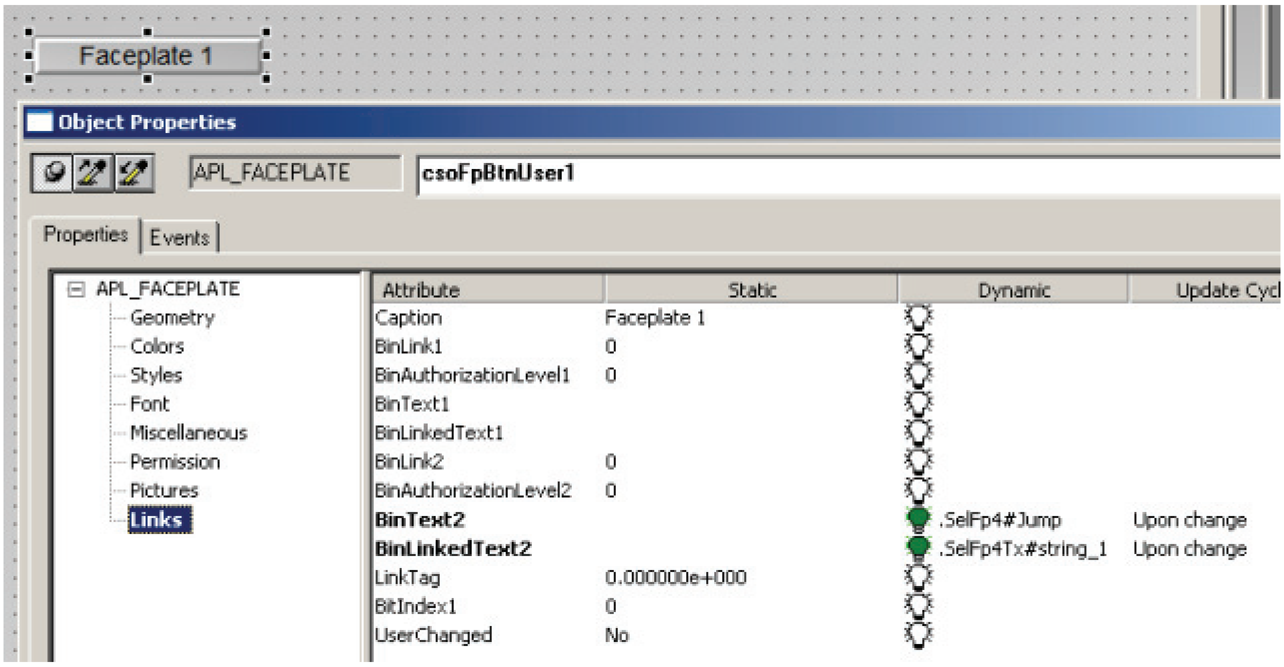
After the parameter with the type BOOL has been created, the following will be possible.

- The parameter can be interconnected exclusively to a parameter of the BOOL type.
- As of PCS7 V7.1 SP1:
  The parameter can be interconnected to an APL structure of the BOOL type.

Option 2:

```
Example script: Anypointer with additional parameter for the text

SelFp4    {BLK_Jump:='1';S7_visible:='false'} :ANY; //Select Faceplate 4
SelFp4Tx  {S7_m_c:='true';
          S7_string_1:='Jump to xxx'
          }
          : BOOL; //Select Faceplate 4 text
```

1. Leave the Anypointer unchanged and assign the text for the labeling of the jump button to an additional parameter such as "SelFp3Tx", as shown in the example script.
2. Adapt the user object "csoFpBtnUser1" as shown below:



Attribute	Static	Dynamic	Update Cycl
Caption	Faceplate 1		
BinLink1	0		
BinAuthorizationLevel1	0		
BinText1			
BinLinkedText1			
BinLink2	0		
BinAuthorizationLevel2	0		
BinText2		.SelFp4#Jump	Upon change
BinLinkedText2		.SelFp4Tx#string_1	Upon change
LinkTag	0.000000e+000		
BitIndex1	0		
UserChanged	No		

Figure 6-4 SelFp1_5

3. Change the following line in the script connected to the attribute "BinText2":

```
Original script
szCaption = GetPropChar(pszParent, "BlockiconCollection",
"UserButtonText1");
```

Changed line:

```
Changed script
pszCaption = GetPropChar(lpszPictureName, lpszObjectName,
"BinLinkedText2");
```

6.2.14 Release for maintenance

The maintenance release serves as information on a process tag at which maintenance, service or calibration needs to be carried out.
For additional information, refer to the APL online help.

You need the "MS_RelOp" input parameter at the AS block. In the "Manual" operating mode the block has to write the "MS_RelOp" input parameter to the "MS_Release" output.

Example script: OutOfService_1

```
VAR_INPUT
     MS_RelOp {S7_m_c := 'true'} :BOOL := FALSE;
END_VAR
VAR_OUTPUT
     MS_Release :DigVal;
END_VAR
Begin
     MS_Release.Value := MS_RelOp AND ManAct.Value;
     IF NOT ManAct.Value THEN
                           MS_RelOp := false;
END_IF;
Status1Bits.x31 := MS_RelOp
```

Note

Make the following settings and interconnections:

- Create the operation enable for operation at the corresponding bit in the "OS_Perm" parameter.
- Link the information from "MS_Release" to the corresponding status word.

Display and operation

- Copy the following objects from the template to display and operate the "Switch maintenance release on/off" function in the faceplate. Add these objects to your faceplate view.
  - APL_OP_BUTTON (@PCS7ElementsAPL.pdl; Open binary operation area; Normal)
  - APL_MULTI_TEXT2 (@PCS7ElementsAPL.pdl; Status display text; Text display)

  For additional information refer to the chapter "Binary display with text boxes (Page 116)".
- Copy the following object from the template for the analog value "PV". Add this object to your faceplate view:
  - APL_ANALOG_OP_DISPLAY (@PCS7ElementsAPL.pdl; Display analog Value; Setpoint)

  For additional information refer to the chapter "Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)".

Note

Ensure here that the corresponding status word / status bit for "MS_RelOp" is interconnected with the operator control enable ("LinkTag3" property) instead of "OS_PermLog".

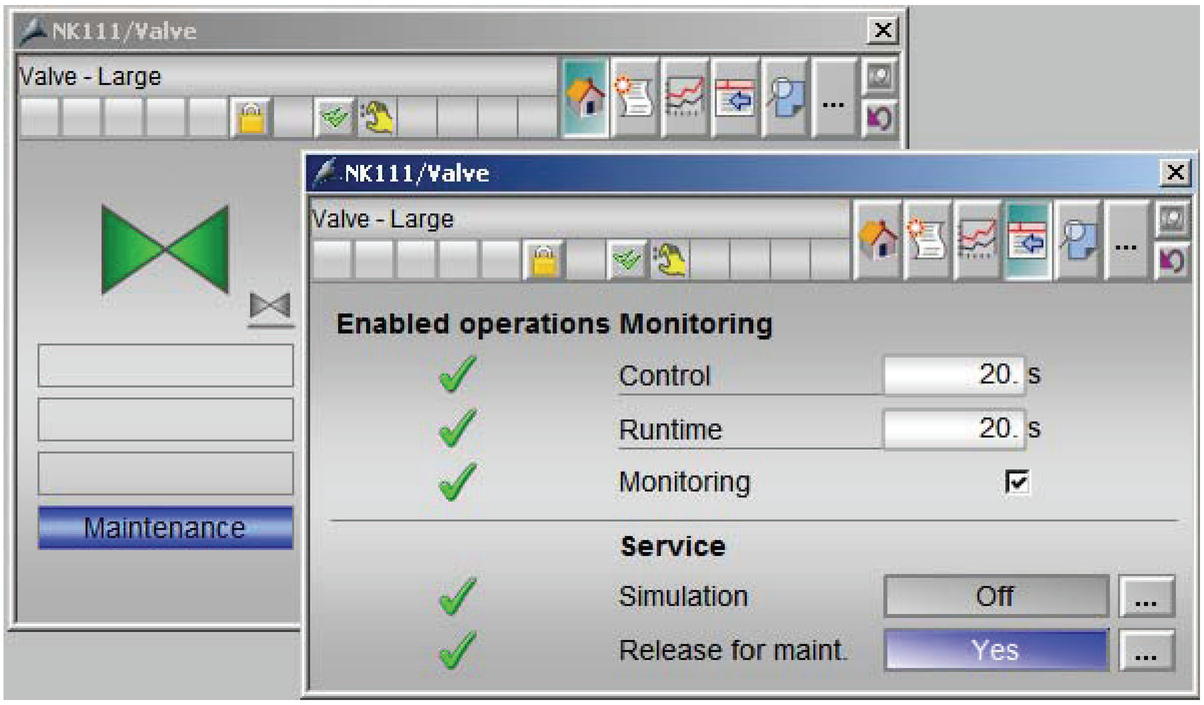


Figure 6-5 Graphic MS_Release1

6.3 Creating block icons

6.3.1 Adapting block icons (overview)

Creating block icons

Function	Section
Creating block icons	• Preparation (Page 82) • Configuring the operating dialog – Calling up the operating dialog directly (Page 83) – Checking the operator authorization before calling up the operating dialog (Page 84)

The fundamental process that you use to create block icons in conformance with APL is the same as the process for creating block icons of the standard library.

Procedure

1. Copy a block icon similar to your requirements from the "@PCS7TypicalsAPLV8.PDL" template screen into your own screen (for example, @PCS7TypicalsMyLib.pdl).
2. Assign parameters to the attributes "type" and "Servername" as reference to the AS block and to the faceplate you want to open.
3. Select the menu command **User object > Edit** in the shortcut menu.

   You can add new objects to the block icon or move existing objects, etc.
4. Assign "speaking" object names to easily recognize the meaning of the objects in the configuration dialog.

   Note

   When the size of the block icon changes, you will have to adapt the size of the internal object "PolylineIcon" because it represents the border of the icon for "HighLightBlockIcon" .

5. Select all objects that are part of the block icon and terminate the "Edit" mode.
6. In the configuration dialog apply the attributes of the new objects you want to dynamize to the external interface of the block icon.
   – If several status displays access the same status word, combine them to one property at the block icon.
7. Assign "speaking" names for the new attributes of the block icon.

   Note

   Always use the same text for the display name and the name of the attribute.

8. Exit the configuration dialog.
9. Connect the dynamics with the new attributes. If necessary, adapt any existing dynamics from the copy template.

See also

Template screen for block icons (Page 15)

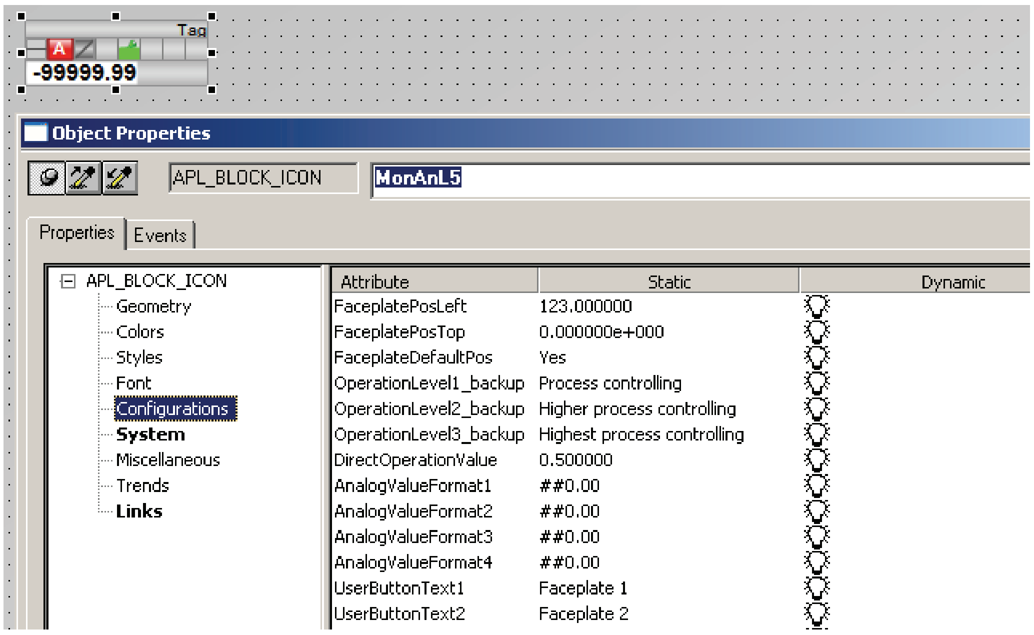
Block icon structure (Page 15)

6.3.2 Description of general properties

6.3.2.1 "Configurations" property group

You configure instance-specific settings in the "Configurations" property group.

"Configurations" property group



APL_BLOCK_ICON MonAnL5

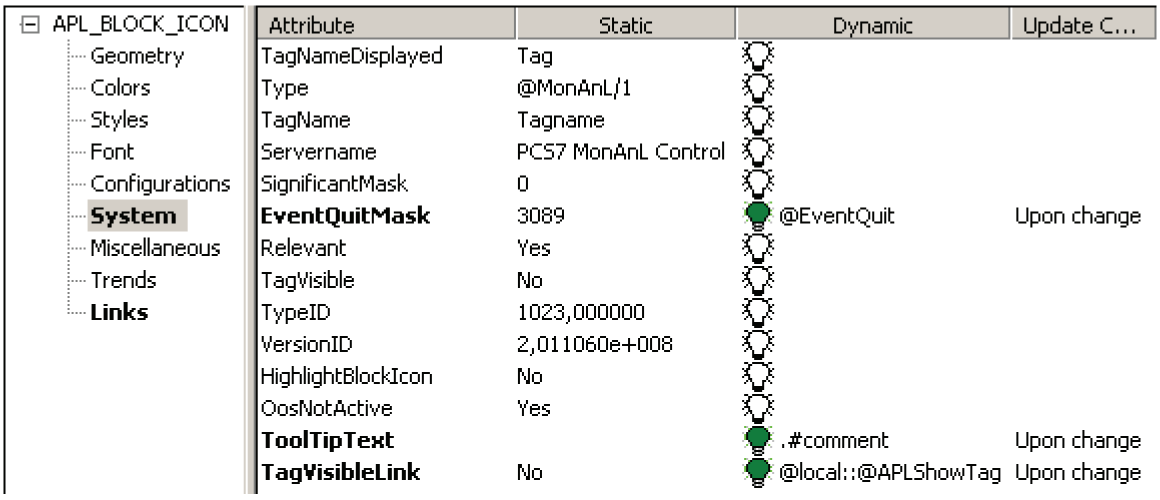
Attribute	Static	Dynamic
FaceplatePosLeft	123.000000	
FaceplatePosTop	0.000000e+000	
FaceplateDefaultPos	Yes	
OperationLevel1_backup	Process controlling	
OperationLevel2_backup	Higher process controlling	
OperationLevel3_backup	Highest process controlling	
DirectOperationValue	0.500000	
AnalogValueFormat1	##0.00	
AnalogValueFormat2	##0.00	
AnalogValueFormat3	##0.00	
AnalogValueFormat4	##0.00	
UserButtonText1	Faceplate 1	
UserButtonText2	Faceplate 2	

Attribute	Meaning
FaceplatesDefaultPos	"FaceplatesDefaultPos" = `No` lets you specify the position of the faceplate you want to open.
OperationLevel1 ... OperationLevel3	"OperationLevel1 ... 3" is used for the instance-specific setting of the operator authorization levels. You can choose between three operator authorizations: The assignment of the three levels to the individual operations will be stored in the faceplate.
DirectOperationValue	DirectOperationValue is a property of the block icon. You specify the percentage values of the two internal buttons for direct operation with the DirectOperationValue. The two outer buttons are automatically determined with DirectOperationValue times the factor 5.
AnalogValueFormat1 … AnalogValueFormat4	AnalogValueFormat1 … 4 is used to set the number format of the block icon and the faceplate.
UserButtonText1 UserButtonText2	UserButtonText1..2 is used to assign parameters to the instance-specific texts for the buttons you use to open other faceplates (jump buttons).

6.3.2.2 "System" property group

The "System" property group contains the basic data of the faceplate.

"System" property group



- APL_BLOCK_ICON
  - Geometry
  - Colors
  - Styles
  - Font
  - Configurations
  - **System**
  - Miscellaneous
  - Trends
  - **Links**

Attribute	Static	Dynamic	Update C...
TagNameDisplayed	Tag	○	
Type	@MonAnL/1	○	
TagName	Tagname	○	
Servername	PCS7 MonAnL Control	○	
SignificantMask	0	○	
EventQuitMask	3089	● @EventQuit	Upon change
Relevant	Yes	○	
TagVisible	No	○	
TypeID	1023,000000	○	
VersionID	2,011060e+008	○	
HighlightBlockIcon	No	○	
OosNotActive	Yes	○	
ToolTipText		● .#comment	Upon change
TagVisibleLink	No	● @local::@APLShowTag	Upon change

Attribute	Meaning
TagNameDisplayed	"TagNameDisplayed" is the tag name shown in the block icon. In the plant hierarchy, you can specify whether the full OS tag name or only parts of it are to be shown in the faceplate.
Type	"Type" is the reference used in the "@PCS7TypicalsAPLV8.PDL" template screen for the "MonAnL" block type of this block icon in version 1. Enter the name of the project-specific FB, for example "MyAPL_BL".
Tagname	"Tagname" contains the actual OS tag name that is sent to the faceplate.
Servername	"Servername" contains the corresponding faceplate type. Enter the name of the project-specific faceplate type, for example ""PCS7 MyAPLFP Control"".
TypeID	TypeID up to and including 1999 is reserved for the APL.

Additional information on these attributes is available in the "PCS 7 Programming Instructions for Blocks".

Attributes of the group display available on the block icon:

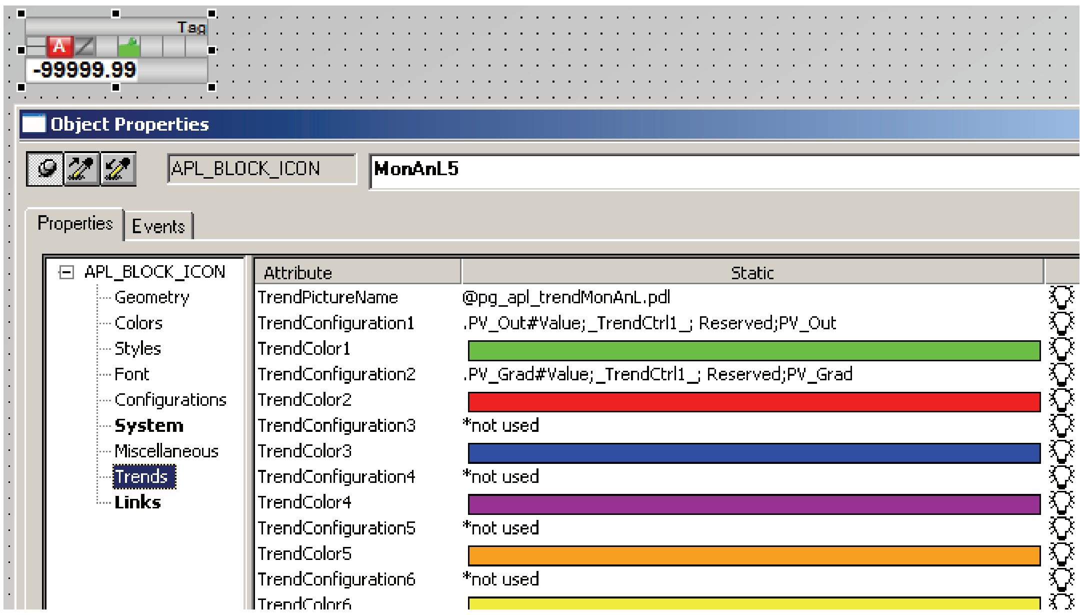
- New attributes of the group display leading outwards:
  - "EventQuitMask" must be interconnected to the "@EventQuit" tag. This setting is necessary because of the adjustable behavior of the "Tolerance" message class in the OS project editor.
  - "SignificantMask" may not be changed.

Attribute	Meaning
TypeID VersionID	• "TypeID" is an assignment number for the APL block icons. Recommendation: Assign your own number for your own icons. The numbers up to an including 2000 are reserved for the APL. • "VersionID" is used for the versioning of the user objects. The version is stored in the form of the date. The display "2.011060e+008" shown in the "BlockIcon2" screen can be broken down and interpreted as follows: Double-click the "VersionID". The following number is shown in the input field: 201106012. This number designates the version from June 1, 2011. The last number designates the increment 2.
ToolTipText	"ToolTipText" and tag name are always interconnected to the same internal object. This means that the tag name is always displayed for the "ToolTipText" . The ToolTipText may not have assigned parameters. For additional information on the individual ToolTipText see Display of the block comment in the ToolTipText of the V7 block icons (Page 123).
TagVisibleLink	The different block icons of the APL are available in the following versions: • Version with permanently displayed tagname • Version in which it is possible to hide the tagname (usually as of Version 5 and higher). The "@PCS7TypicalsAPLV8.PDL" template file has a "Show/hide Tag" button. This button acts on the local computer variable "@local::@APLShowTag" through which the tag name can be shown / hidden centrally.

6.3.2.3 "Trends" property group

You configure the trend view in the faceplate of the "Trends" property group.

"Trends" property group



Attribute	Static
TrendPictureName	@pg_apl_trendMonAnL.pdl
TrendConfiguration1	.PV_Out#Value;_TrendCtrl1_; Reserved;PV_Out
TrendColor1	🟩
TrendConfiguration2	.PV_Grad#Value;_TrendCtrl1_; Reserved;PV_Grad
TrendColor2	🟥
TrendConfiguration3	*not used
TrendColor3	🟦
TrendConfiguration4	*not used
TrendColor4	🟪
TrendConfiguration5	*not used
TrendColor5	🟧
TrendConfiguration6	*not used
TrendColor6	🟨

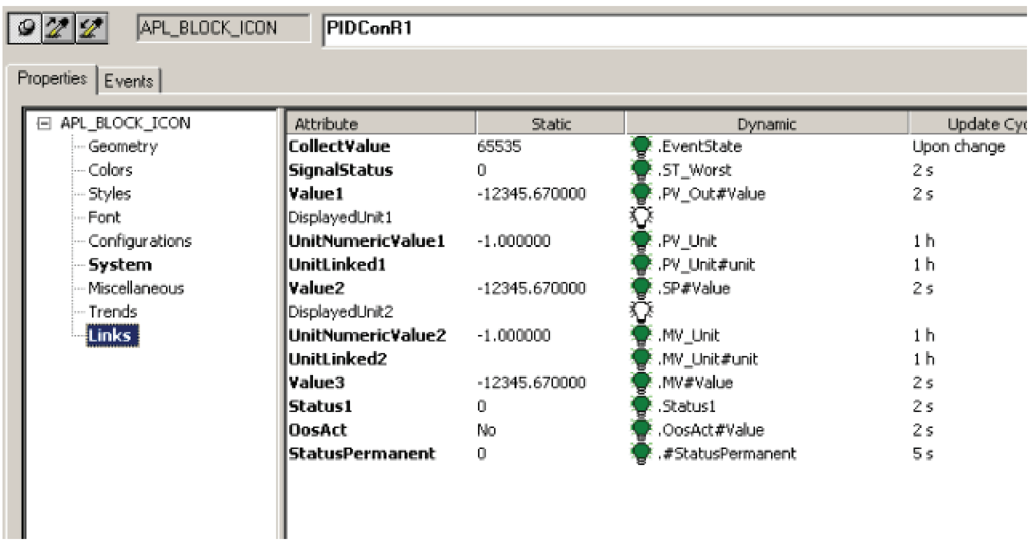
Attribute	Meaning	
TrendPictureName	You use the "TrendPictureName" to select the template screen for the trend view. The figure above includes the "@pg_apl_trendMonAnL.pdl" template screen that contains two trend views: • A trend view for the measured value • A window for the gradient display Use the "@pg_apl_trend.pdl" template screen for the displays of up to 12 trends in a trend view.	
TrendConfiguration1 ... TrendConfiguration12	In "TrendConfiguration1 to 12" you specify which values you want to display. Enter the following information separated by a semicolon:	
	• The variable	for example, ".PV_Out#Value"
	• the TrendControl	always "_TrendCtrl1_"
	• Reserved	always "Reserved"
	• Labeling of the axis and the ruler	"PV_Out"

Attribute	Meaning
	• The archive variable name You must always specify the archive variable names when the PH is changed with an existing trend archive and thus the higher-level designations (HID) of the measuring points. The archive variable name then remains unchanged in the trend archive and can no longer be formed automatically from the HID. In this case the archive variable name has to be specified as the fifth parameter.
TrendColor1 ... TrendColor12	Use TrendColor1 ... 12 to specify the colors of the trends.

6.3.2.4 "Links" property group

You configure dynamic variables in the "Links" property group.

"Links" property group



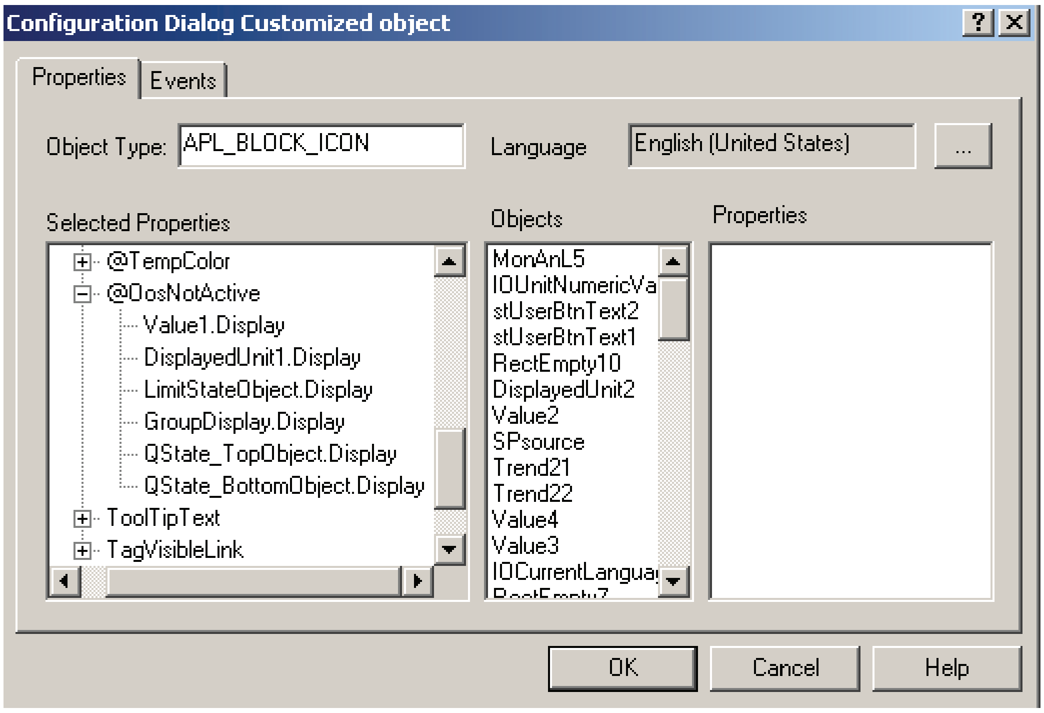
Attribute	Static	Dynamic	Update Cyc
CollectValue	65535	.EventState	Upon change
SignalStatus	0	.ST_Worst	2 s
Value1	-12345.670000	.PV_Out#Value	2 s
DisplayedUnit1			
UnitNumericValue1	-1.000000	.PV_Unit	1 h
UnitLinked1		.PV_Unit#unit	1 h
Value2	-12345.670000	.SP#Value	2 s
DisplayedUnit2			
UnitNumericValue2	-1.000000	.MV_Unit	1 h
UnitLinked2		.MV_Unit#unit	1 h
Value3	-12345.670000	.MV#Value	2 s
Status1	0	.Status1	2 s
OosAct	No	.OosAct#Value	2 s
StatusPermanent	0	.#StatusPermanent	5 s

Attribute	Meaning
SignalStatus	Display of the worst signal status The display of the signal status is implemented using two status displays in the block icon.
DisplayedUnit1 UnitNumericValue1 UnitLinked1	Display of the measurement unit
	"DisplayedUnit1 contains the labeling of the measurement unit
	"UnitNumericValue1" serves to display the measurement unit and is always linked with the integer variable xx_Unit . A script is connected to "UnitNumericValue1". This script has the following function: • At "UnitNumericValue1 = 0" The script selects the text of the S7_unit attribute for the display of the measurement unit. • At "UnitNumericValue1 > 0" The script selects the text for the measurement unit from a table. This table is stored in a resource DLL. You cannot changes this resource DLL. The required text is selected through the integer.
	"UnitLinked1" interconnects with the OS-internal variable that references the WinCC text dictionary ⇒ xx_Unit#unit The S7_xunit attribute is not supported during OS compilation.
Status1	Displays various binary information. "Status1" acts on several internal status displays. Examples are: Operating mode or maintenance status "In progress".
OosAct	Indicates that the block is "Out of operation". The parameter acts on several internal objects. These are switched to invisible at OosAct ="YES" . Take this fact into account when you install new objects you want to display.
StatusPermanent	Shows whether or not a notice is activated.

Switch objects invisible in the "Out of service" operating mode

1. Select the block icon.
2. Use the shortcut menu to open the configuration dialog.
3. Select the "Properties" tab.

   The visible objects of the block icon that are to be switched invisible at a changeover to the operating mode "Out of operation" are listed at the "@OosNotActive" property.


Configuration Dialog Customized object

Properties | Events

Object Type: APL_BLOCK_ICON

Language: English (United States)

Selected Properties:
- @TempColor
- @OosNotActive
  - Value1.Display
  - DisplayedUnit1.Display
  - LimitStateObject.Display
  - GroupDisplay.Display
  - QState_TopObject.Display
  - QState_BottomObject.Display
- ToolTipText
- TagVisibleLink

Objects:
MonAnL5
IOUnitNumericVa
stUserBtnText2
stUserBtnText1
RectEmpty10
DisplayedUnit2
Value2
SPsource
Trend21
Trend22
Value4
Value3
IOCurrentLangua

Properties

Note

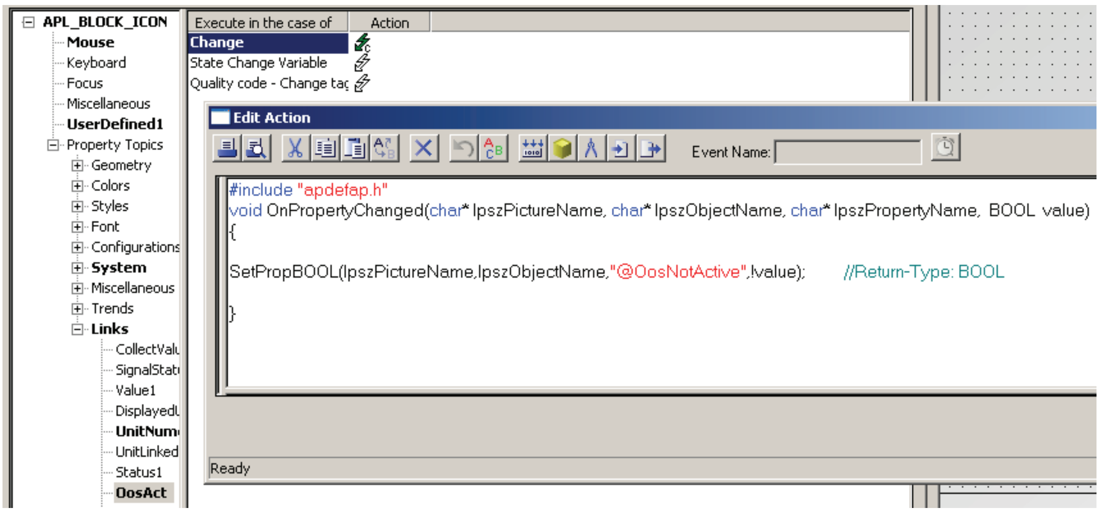
"Display" property for new internally visible objects

Include the "Display" ("Visible") property in the configuration dialog of the user object for new internally visible objects.

Script at the "Change" event

A script is connected to the "Change" event at the "OosAct" attribute. The script evaluates the "@OosNotActive" attribute and controls the visibility of objects.

- If the block is not in the "Out of operation" operating mode, "@OosNotActive" has the value "TRUE "..
- If the block is in the "Out of operation" operating mode, "@OosNotActive" has the value "FALSE"..



```
#include "apdefap.h"
void OnPropertyChanged(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, BOOL value)
{

SetPropBOOL(lpszPictureName,lpszObjectName,"@OosNotActive",!value);        //Return-Type: BOOL

}
```

6.3.3 Operation from the block icon

6.3.3.1 Preparation

The possibility of operation from the block icon was introduced with the APL.

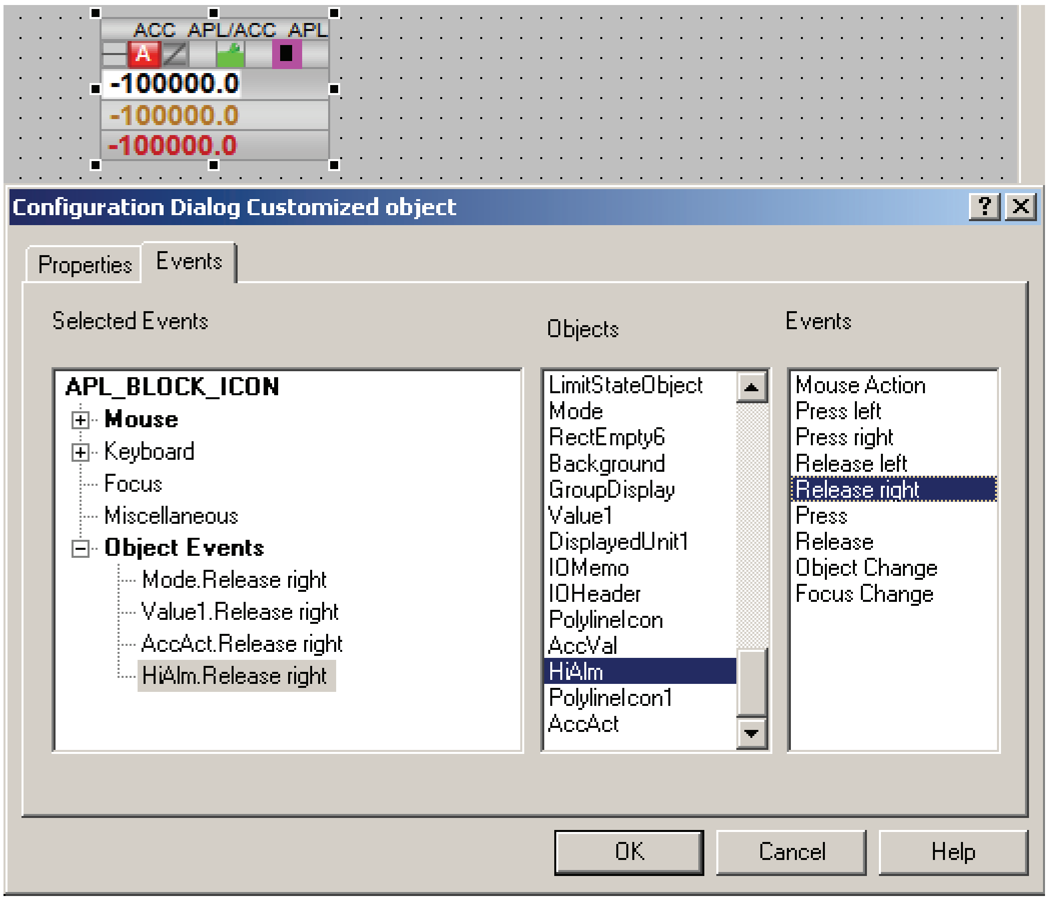
To call up the operating dialog at the block icon, dynamize the event "Right mouse click"(Release Right) with a script.

Making an event available

1. Open the configuration dialog box of the block icon.
2. Select the "Events" tab.
3. In the "Objects" list select the object that you want to operate.

   Example:

   HiAlm, changing an alarm limit
4. Drag the "Release right" event to the Object Events tab.



Selected Events

APL_BLOCK_ICON
- **Mouse**
- Keyboard
- Focus
- Miscellaneous
- **Object Events**
  - Mode.Release right
  - Value1.Release right
  - AccAct.Release right
  - HiAlm.Release right

Objects

LimitStateObject
Mode
RectEmpty6
Background
GroupDisplay
Value1
DisplayedUnit1
IOMemo
IOHeader
PolylineIcon
AccVal
HiAlm
PolylineIcon1
AccAct

Events

Mouse Action
Press left
Press right
Release left
Release right
Press
Release
Object Change
Focus Change

5. Exit the configuration dialog.

Dynamizing an event with script

Dynamize the provided event with a script that calls the corresponding operating dialog. The following section shows the possibilities of implementing the scripts:

- Calling the operating dialog without checking the operator authorization
- Checking the operator authorization before calling up the operating dialog

6.3.3.2 Calling the operating dialog without checking the operator authorization

Recommendation

Use the operation of standard block icons of the APL as the copy template.

Procedure

You can implement the call of the operating dialog without a check of the operator authorization as shown in the following example script:

- Use the transfer parameter "1" for the global script "APL_OpenFaceplate" to specify that the operating dialog is called.
- In the transfer parameter "MFPIC" you specify the view in which the object you want to operate is located.
- The transfer parameter "MFOBJ" identifies the object you want to operate.

Note

The operator authorization does not have to be checked because the check is integrated in the operating dialog.

Example script

Calling the operating dialog without checking the operator authorization

```
#include "apdefap.h"
void OnRButtonUp   (char*lpszPictureName,
                    char*lpszObjectName,
                    char*lpszPropertyName,
                    UINT nFlags,
                    int x,
                    int y)
{
#define MFPIC "@PG_MyAPLFP_Standard.pdl"
#define MFOBJ "@csolOAnaHiAlm"

    APL_OpenFaceplate(lpszPictureName,lpszObjectName,1,MFPIC,MFOBJ);
                                                  //1: MF=OP Dialog
}
```

6.3.3.3 Checking the operator authorization before calling up the operating dialog

You can implement checking of the operator authorization before the call of the operating dialog as shown in the following example script:

- This script checks Bit 2 in the "Status1" status word.
- The operating dialog is only called if Bit 2 is set (block is switched to simulation).

Example script

```
Checking the operator authorization before calling up the operating dialog
#include "apdefap.h"
     void OnRButtonUp
                      (char*lpszPictureName,
                      char*lpszObjectName,
                      char*lpszPropertyName,
                      UINT nFlags,
                      int x, inty)
{
#define BIT(i,status)  (((unsigned long)status & (1<<i))!=0)
#define MFPIC "@PG_PidConL_Standard.pdl"
#define MFOBJ "@csolOAnaPV"

unsigned long value = (unsigned long)GetPropDouble(lpszPictureName,
                      lpszObjectName,"Status1");
                      BOOLxSimuOn=BIT(2,value);
     if (xSimuOn)
     {
       APL_OpenFaceplate(lpszPictureName,lpszObjectName,1,MFPIC,MFOBJ);
     }
}
```

Note

If you use copies of faceplates to create the function, you will have to adapt operations from the block icon present in the script. You have to adapt the scripts because the names of the faceplate view are entered permanently.

Example:

- Source:

  `#define MFPIC "@PG_PidConL_Standard.pdl"`
- Changed to:

  `#define MFPIC "@PG_MyAPLFP_Standard.pdl"`

(see code example above)

6.4 Creating faceplates

6.4.1 Generating a replica from existing faceplates

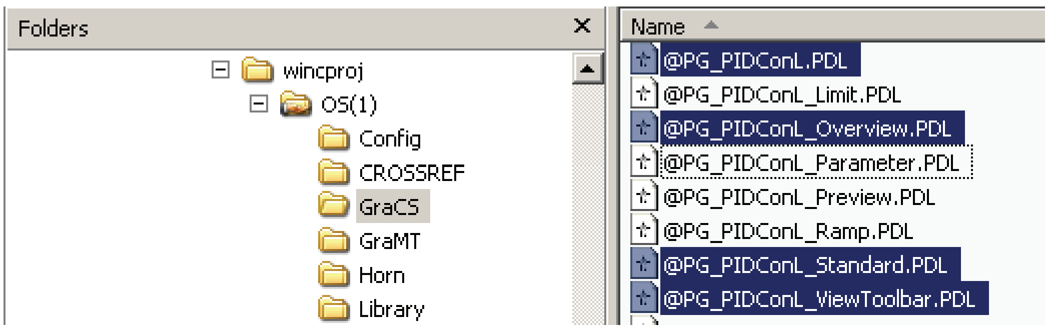
Introduction

The example shows how to create a project-specific version on the basis of the APL standard controller "PIDConL".

Procedure

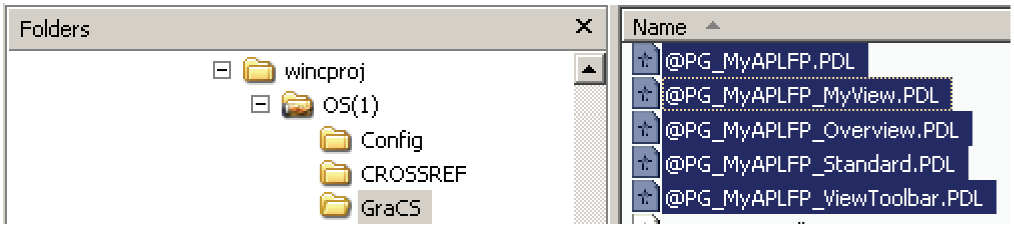
Proceed as follows to create a faceplate in conformance with APL:

1. Copy the files of the suitable faceplate (for example PIDConL) from the templates in the project.



@PG_PIDConL.PDL
@PG_PIDConL_Limit.PDL
@PG_PIDConL_Overview.PDL
@PG_PIDConL_Parameter.PDL
@PG_PIDConL_Preview.PDL
@PG_PIDConL_Ramp.PDL
@PG_PIDConL_Standard.PDL
@PG_PIDConL_ViewToolbar.PDL

2. Save the files under the corresponding name of your block or your faceplate version.



@PG_MyAPLFP.PDL
@PG_MyAPLFP_MyView.PDL
@PG_MyAPLFP_Overview.PDL
@PG_MyAPLFP_Standard.PDL
@PG_MyAPLFP_ViewToolbar.PDL

Views in the example

The following views are used in the example

- Standard view
- Alarm view
- Trend view
- Batch view
- Memo view
- MyView (own created view)

Create a copy of the following template screens:

	Example of a new name for the user faceplate
Basic screen, basis for the faceplate.	@PG_MyAPLFP.PDL
• Overview	@PG_MyAPLFP_Overview.PDL
• View selection	@PG_MyAPLFP_ViewToolbar.PDL
• Standard view	@PG_ MyAPLFP_Standard.PDL
• MyView (own created view)	@PG_MyAPLFP_MyView.PDL

Global views

The following global views exist only once for all faceplate types.

Global view	Global screen
Alarm view	@PG_apl_message.pdl
Trend view	@PG_apl_trend.pdl
Batch view	@PG_apl_batch.pdl
Memo view	@PG_apl_memo.pdl

The views do not require a copy. If required, the global screens are addressed directly from the view selection @PG_MyAPLFP_ViewToolbar.PDL .

6.4.2 Adapting faceplates (overview)

Creating faceplates

Function	Section
Creating replicas	Generating a replica from existing faceplates (Page 85)
Adapting views • Standard view • Alarm view • Trend view • Batch view • Memo view • MyView (own created view)	• Adapting the basic view "@PG_MyAPLFP.PDL" (Page 88) – Screen name in the "Firstview" property – Screen name in the "ToolbarWindow" object – Faceplate type in the "ObjCollection" object in the "BlockType" property • Adapting the overview screen "@PG_MyAPLFP_Overview.PDL" (Page 89) – Interconnections in the AS block or faceplate • Adapting the standard screen "@PG_MyAPLFP_Overview.PDL" (Page 90) – Deleting unused objects in the screen "@PG_MyAPLFP_Standard.pdl" • Adapting the view selection "@PG_MyAPLFP_ViewToolbar.PDL" (Page 91) – Changing the view selection
Assigning operator authorizations	Changing or assigning operator authorizations (Page 93)
Specifying number formats in the faceplate	Specifying number formats for analog values instance-specifically per script (Page 95)
Global settings	Global properties of faceplate objects • Color gradient of the objects (Page 96) • TypeID and VersionID (Page 97) • SelectionBorder (Page 98)
Inserting faceplate objects	• Displaying an analog value (Page 98) (Example: APL_ANALOG_OP_DISPLAY) • Analog value operation (Page 102) with object APL_ANALOG_OP_DISPLAY • Operating a binary value (Page 109) (Example: APL_OP_BUTTON) • Binary value display with text boxes (Page 116) • Bar graphs (Page 106) • Check box (Page 113) • Displaying the signal status (Page 118) • Displaying operating locks (Page 120)
Preparing jumps between faceplates	• Copying buttons and assigning text (Page 67)

6.4.3 Creating a faceplate

6.4.3.1 General specifications in the screen @PG_MyAPLFP.pdl

You have to change properties in the basic screen "@PG_MyAPLFP.PDL".

Note

Saving the basic screen

When saving the basic screen the screen height has to be set to "1".

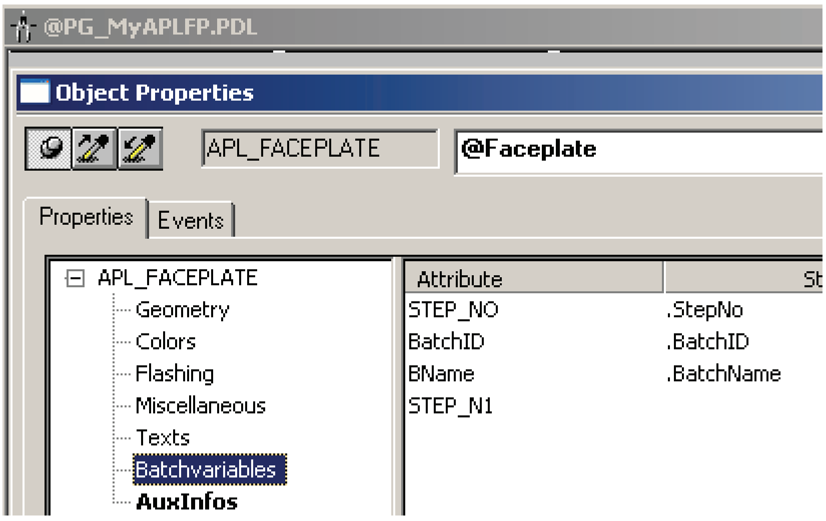
Procedure

The following properties must **always** be set:

1. Enter the new screen name, for example "@PG_MyAPLFP_Standard.PDL" in the "Firstview" property of the "@Faceplate" object.
2. Enter the new screen name, for example "@PG_MyAPLFP_ViewToolbar.PDL" in the "PictureName" property of the "ToolbarWindow" object.
3. Enter the name of the new faceplate type (server name), for example "MyAPLFP" in the "BlockType" property of the "ObjCollection" object.

AS block does not have any variables for SIMATIC BATCH

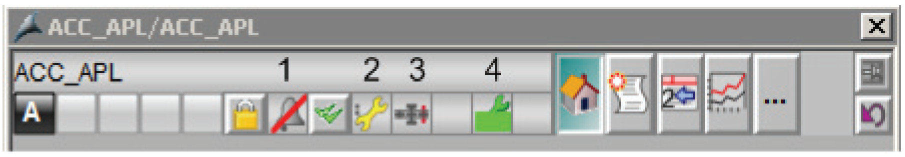
If the AS block does not have any variables for SIMATIC BATCH, remove the corresponding entry in the "@Faceplate" object.



Attribute	St
STEP_NO	.StepNo
BatchID	.BatchID
BName	.BatchName
STEP_N1	

6.4.3.2 General specifications in the screen @PG_MyAPLFP_Overview.pdl

If you use the copy template of the controller (@PG_PIDConL_Overview.pdl) for the overview of the faceplate, adapt the following defaults in the AS block or in the screen (*.PDL):



①	MSGLOCK	Interconnected to parameter STATUS2; BIT0
②	Worst signal status	Interconnected to parameter ST_Worst
③	Batch occupied	Interconnected to parameter STATUS1; BIT0
④	Maintenance status "In progress"	Interconnected to parameter STATUS1; BIT4

Note

Using your own parameter name for the "worst signal status"

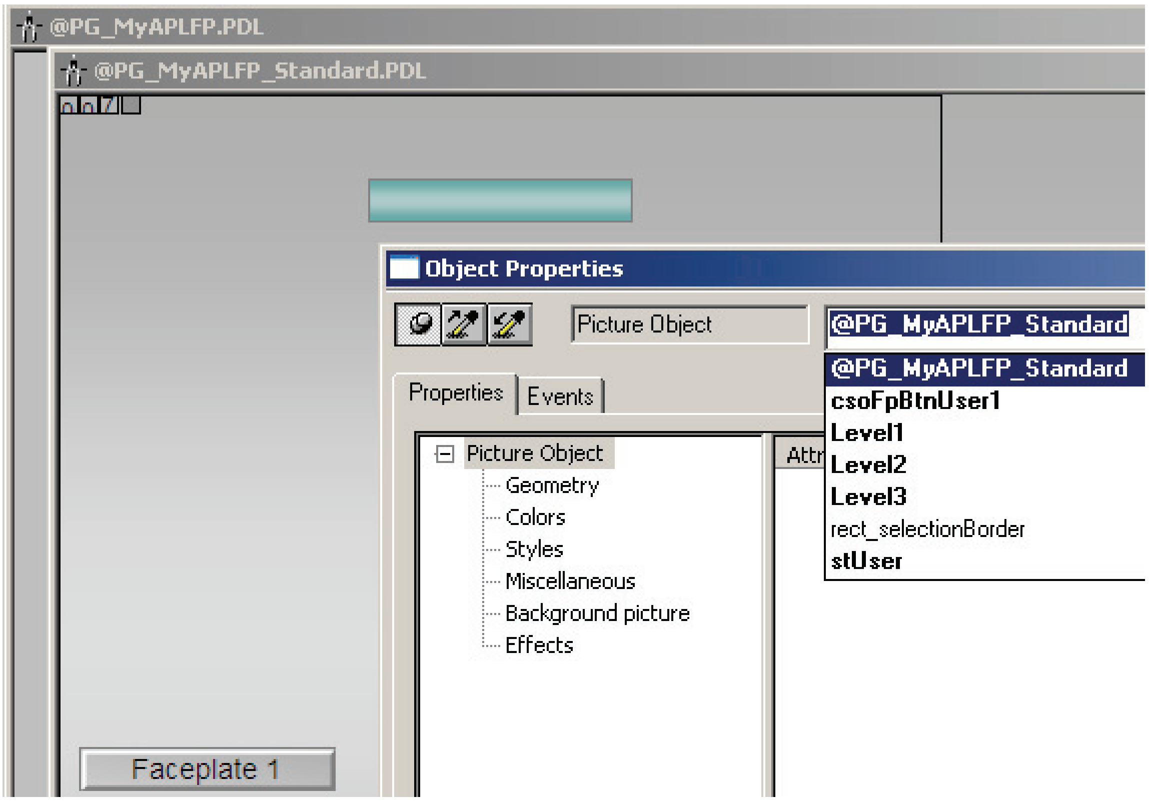
If you replace the parameter name "ST_Worst" with a different parameter name, you will also have to adapt the parameter name in the two status displays lying over each other.

6.4.3.3 General specifications in the screen @PG_MyAPLFP_Standard.pdl

In the copy you will delete all objects except the objects that are shown in the following figure.

The following objects remain unchanged:

- "csoFpBtnUser1" for the jump to the block that is interconnected with SelFp1 .
- "Level1"
- "Level2"
- "Level3"
- "rect_selectionBorder"
- "stUser"



6.4.3.4 Changing the view selection

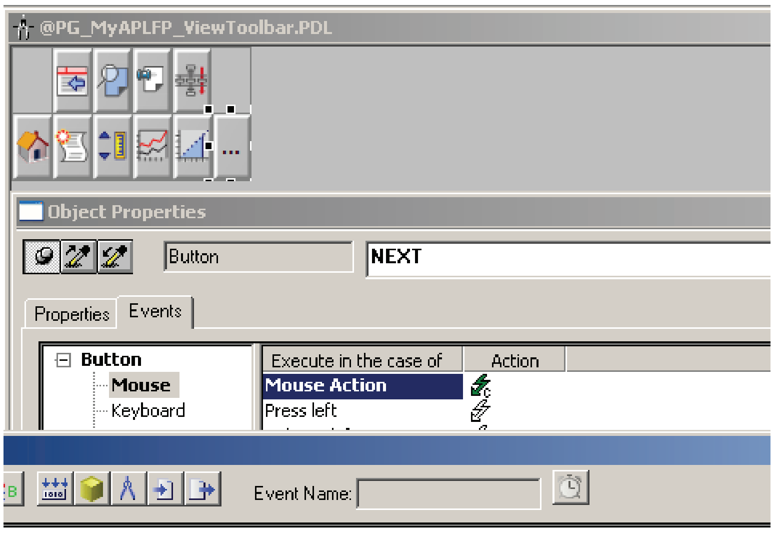
The example screen "@PG_MyAPLFP_ViewToolbar.PDL" with its individual elements is shown below.

The objects located in the foreground have been moved downwards so that the elements for selecting the individual views are shown. The objects lying in the background are shown in the top row.

- Use the "STANDARD" object (bottom left) to select the standard view. This object is always visible and can be selected.

  The "NEXT" object (bottom right) activates a script at "Mouse Click" that switches between the visibility of the top objects and the bottom objects. The "NEXT" object is always visible and can be selected.

The following figure shows the script for switching the visibility of the bottom and top objects:



```
//Button set 1
SetPropBOOL(lpszPictureName,"MESSAGE", "Visible", bButtonSet1);
SetPropBOOL(lpszPictureName,"LIMIT", "Visible", bButtonSet1);
SetPropBOOL(lpszPictureName,"TREND", "Visible", bButtonSet1);
SetPropBOOL(lpszPictureName,"RAMP", "Visible", bButtonSet1);

//Button set 2
SetPropBOOL(lpszPictureName,"PARAMETER", "Visible", bButtonSet2);
SetPropBOOL(lpszPictureName,"PREVIEW", "Visible", bButtonSet2);
SetPropBOOL(lpszPictureName,"MEMO", "Visible", bButtonSet2);
SetPropBOOL(lpszPictureName,"BATCH", "Visible", bButtonSet2);
```

- The object name of a button determines the name of the view that is called.

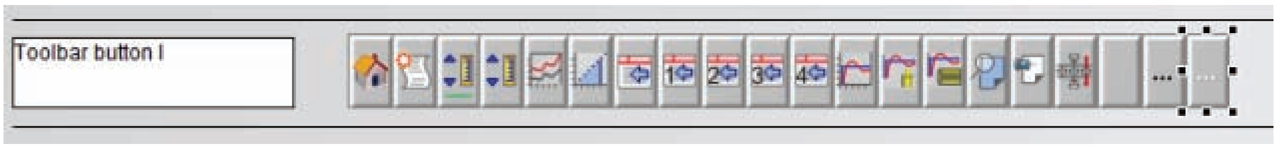
  Example:

  The button with the "Name"=MyView object serves to select the completely new created view "@PG_MyAPLFP_MyView.PDL".

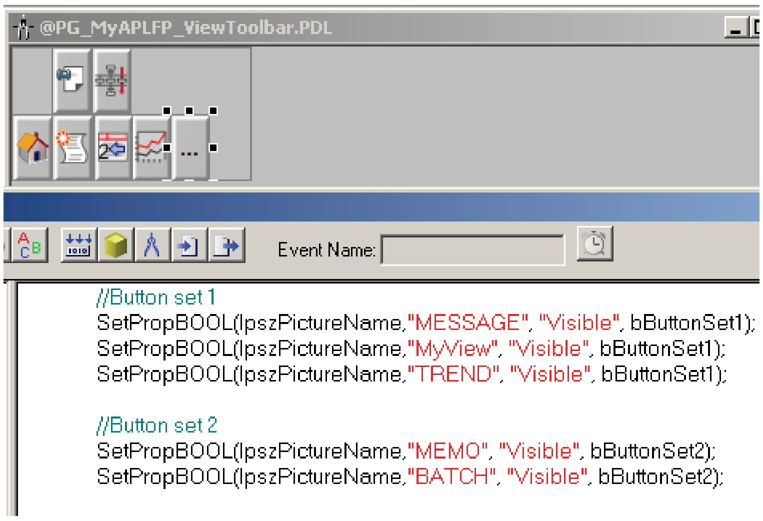
Procedure

- Enter the corresponding text, for example "MyView" at the "ToolTip Text" attribute of the button.
- You can save your own screen for the button at the attributes "PictureStatus On" and "PictureStatus Off".

  The template screen contains a matching selection of buttons:



- You can configure the view selection as shown in the following figure, for example, for the example described in the Style Guide.
  - Adapt the entries at the "NEXT" object in the script.
  - To ensure that the last toolbar representation is displayed when a screen composition is called, adapt the entries in the script at the "Screen selection" event of the "@PG_MyAPLFP_ViewToolbar.PDL" screen.



```
//Button set 1
SetPropBOOL(lpszPictureName,"MESSAGE", "Visible", bButtonSet1);
SetPropBOOL(lpszPictureName,"MyView", "Visible", bButtonSet1);
SetPropBOOL(lpszPictureName,"TREND", "Visible", bButtonSet1);

//Button set 2
SetPropBOOL(lpszPictureName,"MEMO", "Visible", bButtonSet2);
SetPropBOOL(lpszPictureName,"BATCH", "Visible", bButtonSet2);
```

Result

The figure above shows the objects lying in the foreground. These objects show the buttons for selecting the views:

- Standard view
- Alarm view
- MyView (own created view)
- Trend view

Click the "..." button to change over the display. (Next).

The buttons for the selection of the views will be displayed:

- Memo view
- Batch view

6.4.3.5 Changing or assigning operator authorizations

The "stUser" object uses a script to update the operability of the objects listed by you in the screen "@PG_MyAPLFP.pdl" in the following cases:

- At the screen selection
- At a user change (when the faceplates are not closed)
- At changes in the local operator authorization

Interconnecting attributes of operable objects

Also interconnect the following attributes for the operable objects:

- The AS enable (for example "OS_PermLog") the attribute "PermissionTag"
- At the analog displays the attribute "LinkTag3".

Note

An interconnection of the attributes "PermissionTag" or "LinkTag3" is not mandatory if the default for these properties is 16#FFFFFFFF or 4294967295.

Adapting the script

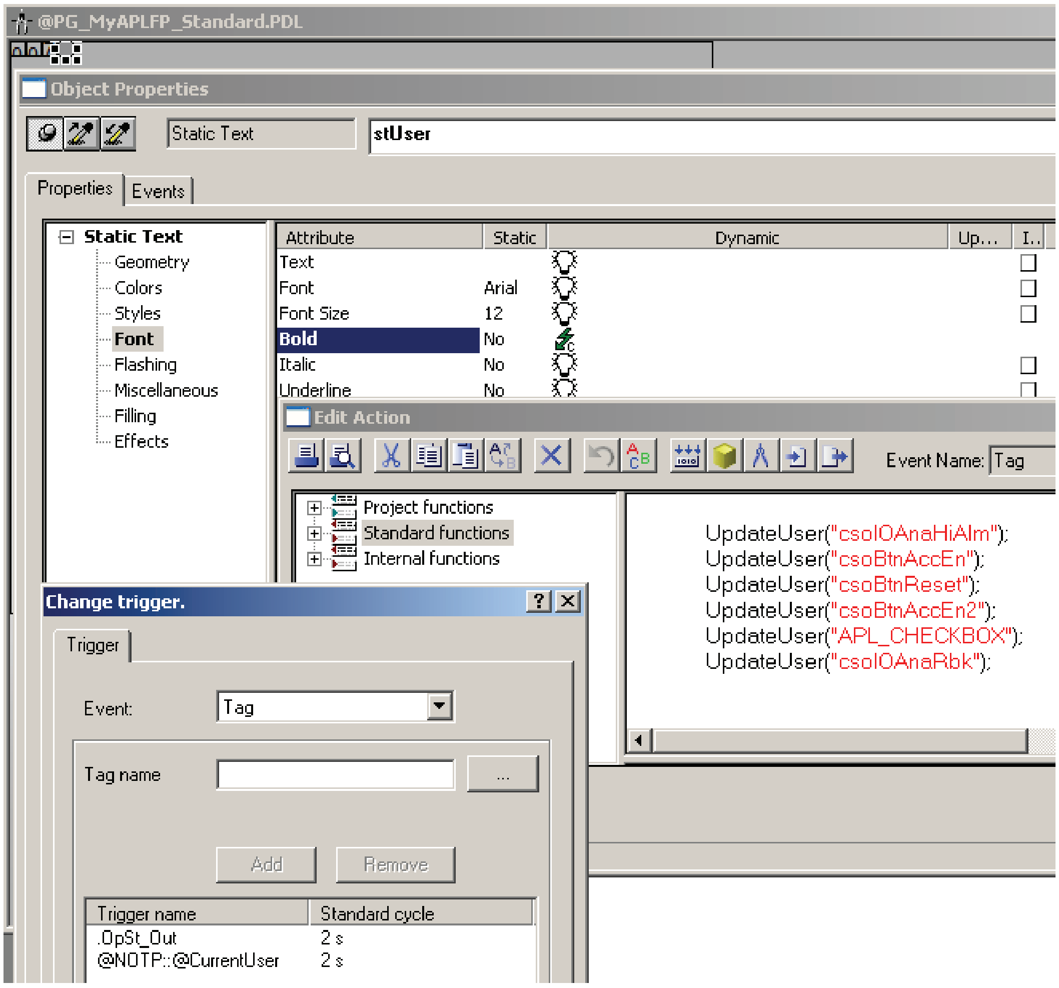
Adapt the script at the "bold" attribute in the "stUser" object as follows:

- List the operable objects in this script.
- Delete unused elements from the original or comment them out.
- Set the operator authorization level directly at the operable objects:
  - At analog value operation the "OperationLevel" attribute
  - At binary value operation the attribute "BinAuthorizationLevel1 .. x"

Operator authorization levels of the objects

- 1 = Process operation
- 2 = Higher order process operation
- 3 = Highest order process operation

The following figure shows the script with the operable objects:



Attribute	Static	Dynamic	Up...	I..
Text				☐
Font	Arial			☐
Font Size	12			☐
Bold	No			
Italic	No			☐
Underline	No			☐

```
UpdateUser("csoIOAnaHiAlm");
UpdateUser("csoBtnAccEn");
UpdateUser("csoBtnReset");
UpdateUser("csoBtnAccEn2");
UpdateUser("APL_CHECKBOX");
UpdateUser("csoIOAnaRbk");
```

Event: Tag

Tag name

Trigger name	Standard cycle
.OpSt_Out	2 s
@NOTP::@CurrentUser	2 s

Additional information

Additional information about the parameter assignment of the AS enable is available in the following chapters:

- Chapter "Binary value operation / PermissionTag"
- Chapter "Analog value operation / Process-related enable".

6.4.3.6 Number formats for analog values

Use the screen selection script of the view in the APL for instance-specific formatting of the display of numbers (decimals and decimal places). The script solution can be used better than direct connections.

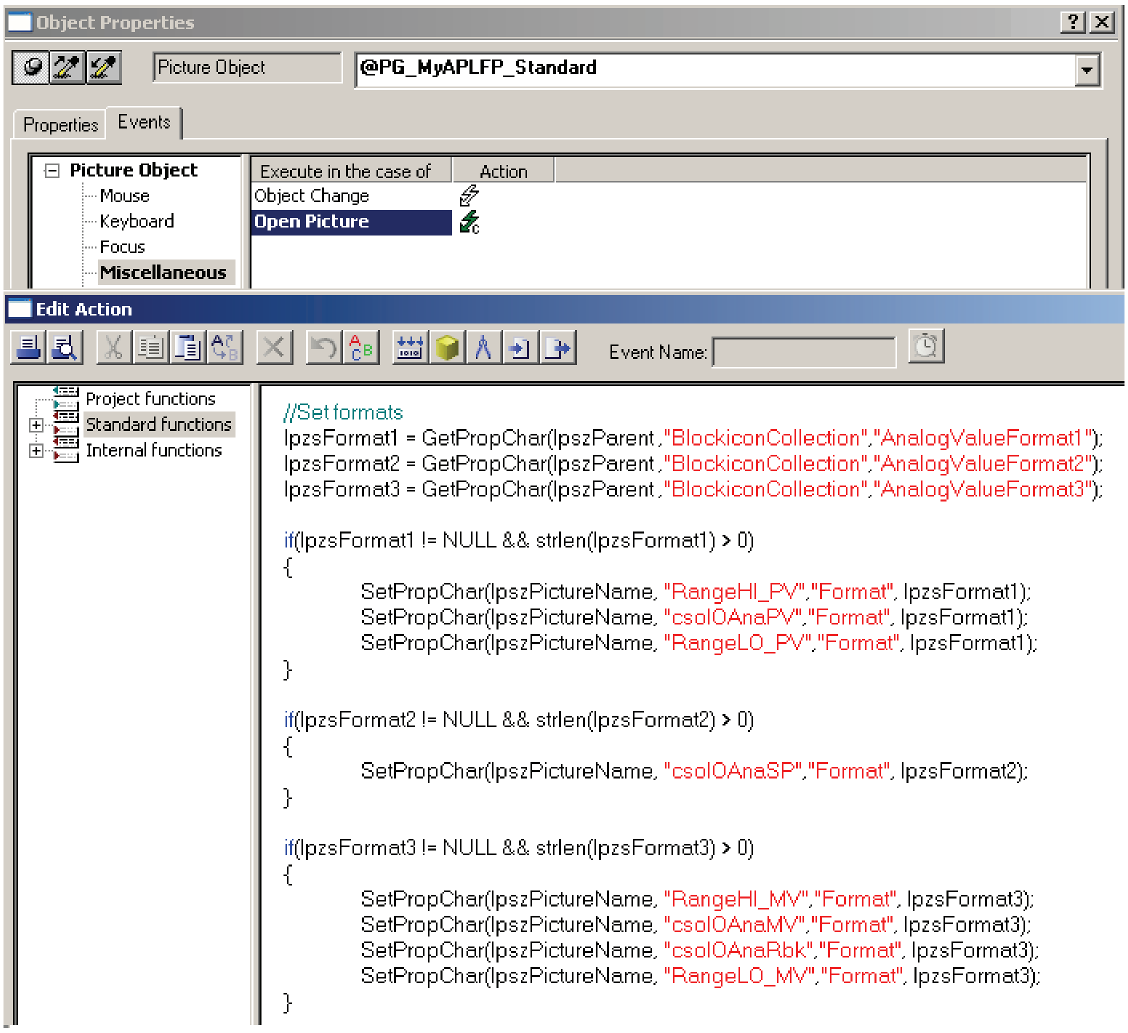
Procedure

Adapt the script for the action at the screen selection attribute as follows:

- Delete unused elements of the original or comment them out.
- Assign analog values that are to be shown in formatted form to the corresponding analog value format in the script, for example "AnalogValueFormat1", "AnalogValueFormat2" or "AnalogValueFormat3".

Example code

The following figure shows the script for formatting analog values:



```
//Set formats
lpzsFormat1 = GetPropChar(lpszParent,"BlockiconCollection","AnalogValueFormat1");
lpzsFormat2 = GetPropChar(lpszParent,"BlockiconCollection","AnalogValueFormat2");
lpzsFormat3 = GetPropChar(lpszParent,"BlockiconCollection","AnalogValueFormat3");

if(lpzsFormat1 != NULL && strlen(lpzsFormat1) > 0)
{
        SetPropChar(lpszPictureName, "RangeHI_PV","Format", lpzsFormat1);
        SetPropChar(lpszPictureName, "csolOAnaPV","Format", lpzsFormat1);
        SetPropChar(lpszPictureName, "RangeLO_PV","Format", lpzsFormat1);
}

if(lpzsFormat2 != NULL && strlen(lpzsFormat2) > 0)
{
        SetPropChar(lpszPictureName, "csolOAnaSP","Format", lpzsFormat2);
}

if(lpzsFormat3 != NULL && strlen(lpzsFormat3) > 0)
{
        SetPropChar(lpszPictureName, "RangeHI_MV","Format", lpzsFormat3);
        SetPropChar(lpszPictureName, "csolOAnaMV","Format", lpzsFormat3);
        SetPropChar(lpszPictureName, "csolOAnaRbk","Format", lpzsFormat3);
        SetPropChar(lpszPictureName, "RangeLO_MV","Format", lpzsFormat3);
}
```

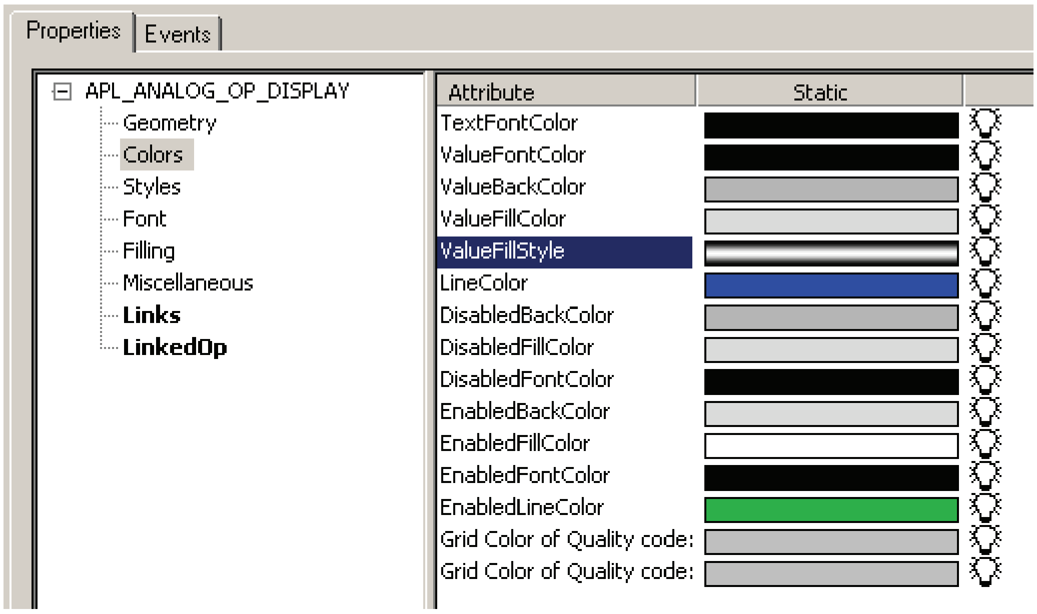
Recommendation for testing the faceplates

Use the Global Script diagnostics to test the created faceplates. This allows you to easily recognize objects that no longer exist but are still referenced in the scripts.

6.4.4 Global properties of faceplate objects

Description of the global properties of user objects in the APL faceplates.

6.4.4.1 Color gradient of the objects



Properties | Events

- APL_ANALOG_OP_DISPLAY
  - Geometry
  - Colors
  - Styles
  - Font
  - Filling
  - Miscellaneous
  - **Links**
  - **LinkedOp**

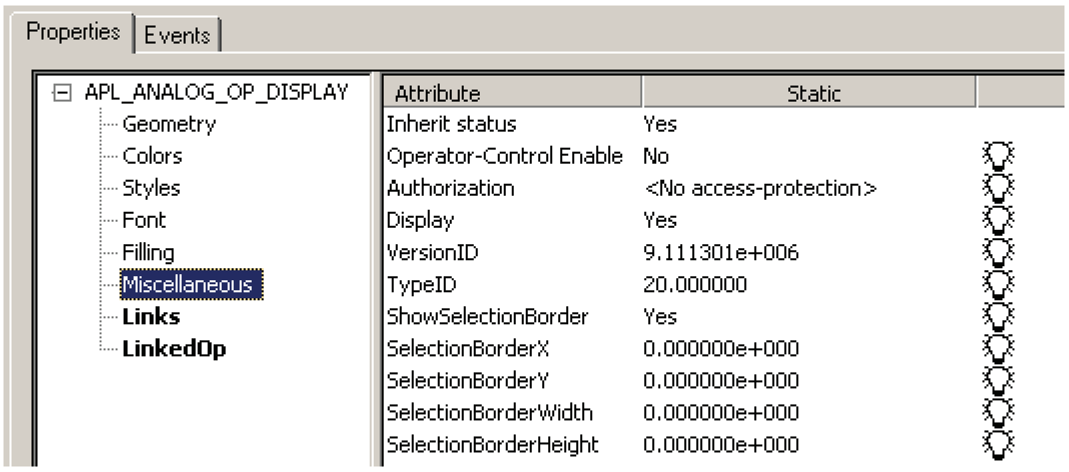
Attribute	Static
TextFontColor	⬛
ValueFontColor	⬛
ValueBackColor	▨
ValueFillColor	⬜
ValueFillStyle	▤
LineColor	🟦
DisabledBackColor	▨
DisabledFillColor	⬜
DisabledFontColor	⬛
EnabledBackColor	⬜
EnabledFillColor	⬜
EnabledFontColor	⬛
EnabledLineColor	🟩
Grid Color of Quality code:	▨
Grid Color of Quality code:	▨

Use ValueFillStyle = 7 at the property depending on the screen object used.
For additional information refer to the chapter "Specifications for color (Page 32)".

6.4.4.2 TypeID and VersionID

Properties of the user objects

The attributes "VersionID" and "TypeID" are properties of the user objects.



Attribute	Static	
Inherit status	Yes	
Operator-Control Enable	No	💡
Authorization	<No access-protection>	💡
Display	Yes	💡
VersionID	9.111301e+006	💡
TypeID	20.000000	💡
ShowSelectionBorder	Yes	💡
SelectionBorderX	0.000000e+000	💡
SelectionBorderY	0.000000e+000	💡
SelectionBorderWidth	0.000000e+000	💡
SelectionBorderHeight	0.000000e+000	💡

Attribute "VersionID"

The attribute "VersionID" is used to version the user objects.

- The version is stored here in form of the date.
- The representation is shown in the previous figure: 9.111301+006 means 13/11/2009.

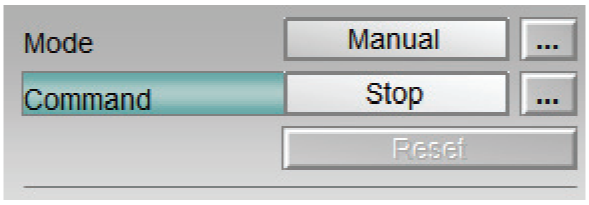
Attribute "TypeID"

The "TypeID" attribute is an assignment number for user objects that are used in the APL faceplates.

- The "TypeID" can remain unchanged if you configure only unchanged APL user objects in your structure.
- If you change objects, for example, by including new internal objects, you should specify a new "TypeID". The new "TypeID" must be **> 2500** , because the
- TypeID up to and including 1999 is reserved for the APL.

6.4.4.3 ShowSelectionBorder

Use the attribute "ShowSelectionBorder"= `Yes` to give an operable function a color marking when it is selected.



Specifications for APL standard objects

With APL standard objects the marking is entered at a fixed position relative to the operated object if the following conditions are met:

- The size remains unchanged.
- The standard value "154" is assigned to the "Position X" attribute of the object you want to select.
- The value 0 is assigned to the following subordinate properties:
  - "SelectionBorderX"
  - "SelectionBorderY"
  - "SelectionBorderWidth"
  - "SelectionBorderHeight"

Individual design

You configure individual positions or sizes for the color marking by using the attributes that are specified in the section "Specifications for APL standard objects".

6.4.5 Displaying and operating analog values

6.4.5.1 APL_ANALOG_OP_DISPLAY analog value display

The same object is used for an analog value display and for an analog value operation. The procedure differs in the parameter assignment and the variable connection.

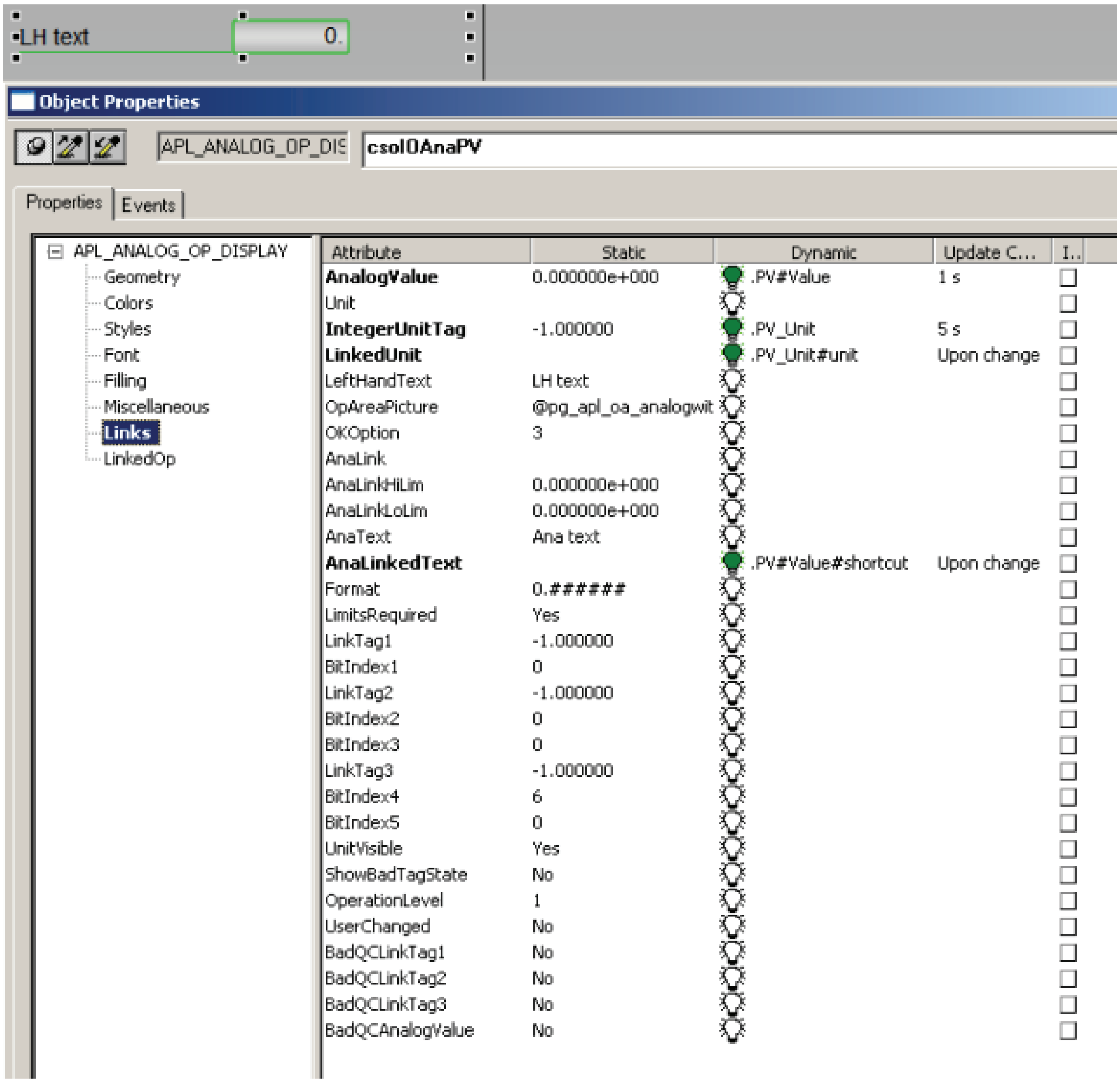
Assigning parameters to the analog value display

1. Copy the APL_ANALOG_OP_DISPLAY(@PCS7ElementsAPL.pdl; Display analog Value; Setpoint) object from the template screen and insert the object into your faceplate view.

2. Enter the object name in the screen selection script for a number format assignment.

   For additional information refer to the chapter "Number formats for analog values (Page 95)".

   For information on positioning refer to the chapter "Specifications for faceplates (Page 17)".



Attribute	Static	Dynamic	Update C...	I..
AnalogValue	0.000000e+000	● .PV#Value	1 s	☐
Unit				☐
IntegerUnitTag	-1.000000	● .PV_Unit	5 s	☐
LinkedUnit		● .PV_Unit#unit	Upon change	☐
LeftHandText	LH text			☐
OpAreaPicture	@pg_apl_oa_analogwit			☐
OKOption	3			☐
AnaLink				☐
AnaLinkHiLim	0.000000e+000			☐
AnaLinkLoLim	0.000000e+000			☐
AnaText	Ana text			☐
AnaLinkedText		● .PV#Value#shortcut	Upon change	☐
Format	0.######			☐
LimitsRequired	Yes			☐
LinkTag1	-1.000000			☐
BitIndex1	0			☐
LinkTag2	-1.000000			☐
BitIndex2	0			☐
BitIndex3	0			☐
LinkTag3	-1.000000			☐
BitIndex4	6			☐
BitIndex5	0			☐
UnitVisible	Yes			☐
ShowBadTagState	No			☐
OperationLevel	1			☐
UserChanged	No			☐
BadQCLinkTag1	No			☐
BadQCLinkTag2	No			☐
BadQCLinkTag3	No			☐
BadQCAnalogValue	No			☐

Brief description of the attributes relevant for the analog value display.

Detailed description will follow.

Attribute	Brief description
Colors > LineColor	Color margin of the analog value display
Links > AnalogValue	Display of the actual analog value
Links > Unit	Display of the unit
Links > IntegerUnitTag	Source of the unit is determined by means of the number and a table **Note:** Information about this table with measurement units in accordance with the specification "PA-Profile for Process Device" from "PROFIBUS and PROFINET International" is available in the APL online help.
Links > LinkedUnit	Source of the unit through S7_Unit attribute if number = 0
Links > LeftHandText	Labeling of the display if no S7_shortcut is used
Links > AnaLinkedText	Labeling of the display with S7_shortcut instance-specifically
Links > UnitVisible	Hiding of the measurement unit

Interconnecting the analog value

- The actual analog value is interconnected with the "AnalogValue" attribute.
- You configure the representation of the color margin of the analog value display with the "LineColor" attribute.

Representing the measurement unit

There are three attributes in the object to display the measurement unit.

Attribute	Meaning
Unit	"Unit" is the displayed text for the measurement unit
IntegerUnitTag	"IntegerUnitTag" is interconnected with the integer variable, for example, "PV_Unit".
LinkedUnit	"LinkedUnit" is interconnected with the OS-internal variable "PV_Unit#unit" that references the text dictionary.

Script at the "IntegerUnitTag" event

A script that carries out the following actions is interconnected in the event of "IntegerUnitTag":

- IntegerUnitTag `= 0`; the script transfers the text of the "`S7_unit`" attribute.
- IntegerUnitTag `> 0`; the script searches for the corresponding text in the measurement unit table using the integer value. This table with standardized units will be stored in the OS.

Note

The unit attribute "`S7_xunit`" is not taken into consideration during compilation of the OS, if it has been created in the SCL source in a structure.

No OS-internal variable is created either.

Use the "UnitVisible" attribute to suppress the display of the measurement unit.

Deleting unused dynamics

You will have to delete the remaining dynamics at the following attributes:

- "AnaLinkHiLim"
- "AnaLinkLo"
- "LimLinkOp"
- "LinkTag3"

  Parameterize `LinkTag3=0`. The object can then not be operated.

Labeling the analog value display

You can interconnect the `S7_Shortcut` attribute to the "AnaLinkedText" attribute and thus create an instance-specific text, for example "PV#Value#shortcut".

In the case of OS-internal variables it is advisable to always set the Trigger variable to "Change".

- If no text is entered in the `S7_Shortcut` , the text entered under the "LeftHandText" attribute is displayed as the labeling.
- If a text is entered in the `S7_Shortcut` , it will be displayed as the labeling.

For information about the ""AnaText" " property refer to the chapter "Analog value operation with object APL_ANALOG_OP_DISPLAY (Page 102)".

See also

Configuring the highlighting of analog value displays (Page 23)

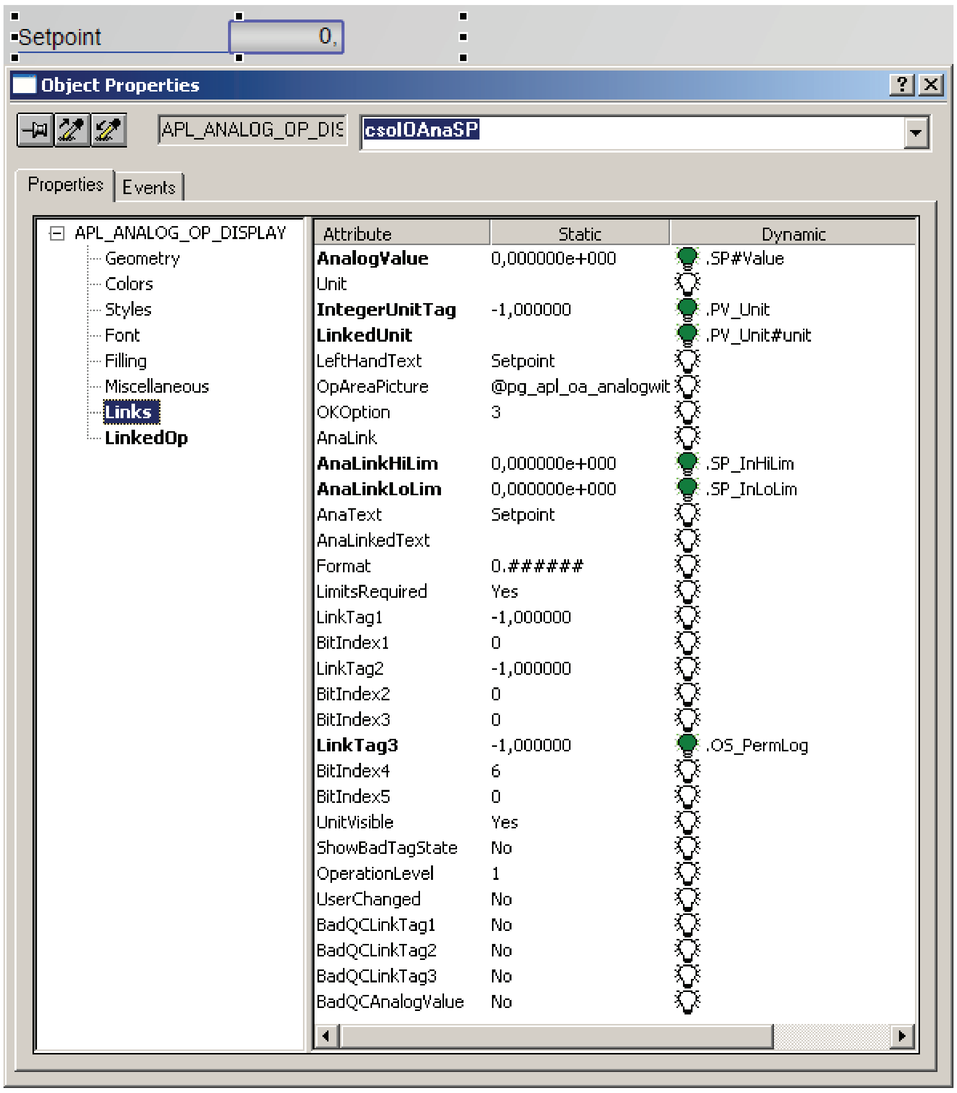
6.4.5.2 Analog value operation with object APL_ANALOG_OP_DISPLAY

The following section describes the parameter assignment for an analog value operation.

Procedure

1. Copy the APL_ANALOG_OP_DISPLAY (@PCS7ElementsAPL.pdl; Display analog Value; Setpoint) object from the template screen and insert the object into your faceplate view.

   For information on positioning please refer to " ShowSelectionBorder (Page 98)".
2. Enter the object name in the screen selection script for a number format assignment. You can find more information on this in the chapter "Number formats for analog values (Page 95)"
3. Enter the object name in the "stUser" object of your faceplate for an update of the operator authorization when the user changes.



Attribute	Static	Dynamic
AnalogValue	0,000000e+000	.SP#Value
Unit		
IntegerUnitTag	-1,000000	.PV_Unit
LinkedUnit		.PV_Unit#unit
LeftHandText	Setpoint	
OpAreaPicture	@pg_apl_oa_analogwit	
OKOption	3	
AnaLink		
AnaLinkHiLim	0,000000e+000	.SP_InHiLim
AnaLinkLoLim	0,000000e+000	.SP_InLoLim
AnaText	Setpoint	
AnaLinkedText		
Format	0.######	
LimitsRequired	Yes	
LinkTag1	-1,000000	
BitIndex1	0	
LinkTag2	-1,000000	
BitIndex2	0	
BitIndex3	0	
LinkTag3	-1,000000	.OS_PermLog
BitIndex4	6	
BitIndex5	0	
UnitVisible	Yes	
ShowBadTagState	No	
OperationLevel	1	
UserChanged	No	
BadQCLinkTag1	No	
BadQCLinkTag2	No	
BadQCLinkTag3	No	
BadQCAnalogValue	No	

Brief description of the attributes relevant for the analog value operation

Detailed description will follow.

Attribute	Brief description
Links > AnalogValue	Display of the actual analog value
Links > Unit	Display of the unit
Links > IntegerUnitTag	Source of the unit is determined by means of the number and a table **Note:** Information about this table with measurement units in accordance with the specification "PA-Profile for Process Device" from "PROFIBUS and PROFINET International" is available in the APL online help.
Links > LinkedUnit	Source of the unit through S7_Unit attribute if number = 0
Links > LeftHandText	Labeling of the display if no S7_shortcut is used
Links > OpAreaPicture	Selection of the operating area (for example with or without slider)
Links > OkOption	Evaluation of the input value after execution OK
Links > AnaLinkHiLim	Measuring range end for slider and incremental adjustment
Links > AnaLinkHiLim	Measuring range start for slider and incremental adjustment
Links > AnaText	Text for the operation log
Links > AnaLinkedText	Labeling of the display with S7_shortcut instance-specifically
Links > Format	Number format for analog value you want to display
Links > LinkTag3	Process-related operator authorization AS variable
Links > BitIndex4 Bit	Address in the word for the process-related operator authorization.
Links > UnitVisible	Hiding of the measurement unit
Links > OperationLevel	Configuring the operator authorization levels
LinkedOp > LinkOp	Analog value you want to operate

Configuring the analog value display

The analog value you want to display is interconnected with the "AnalogValue" attribute.

Operator control of analog value

The parameter you want to operate is interconnected with the "LinkOp" attribute. If this is the same parameter as the value you want to display, then interconnect the same parameter again.

For information about the measurement unit, labeling of the display and formatting refer to the section "APL_ANALOG_OP_DISPLAY analog value display (Page 98)".

Operating limits

The "AnaLinkHiLim" and "AnaLinkLoLim" properties are used to specify the operating limits of the analog value operation. Decisive for the slider and the incremental operation.

Text for the operation log

Enter the text for the operation log in the "AnaText" property. An entry must be made here, otherwise the operated parameter will be missing in the operation log.

Process-related enable

Use the "LinkTag3" attribute to interconnect the process-related enable for the operation.

- **Recommendation**:

  Use the "OS_PermLog" parameter.

  Use "BitIndex4" to assign parameters to the bit in the double word (starting from Bit0) that contains the enable.
- The following is required if you do not want to connect a variable to the "LinkTag3" attribute:
  – Parameterize the value in "LinkTag3" from -1 to +1.
  – Address the Bit0 with "BitIndex4".

User-specific enable

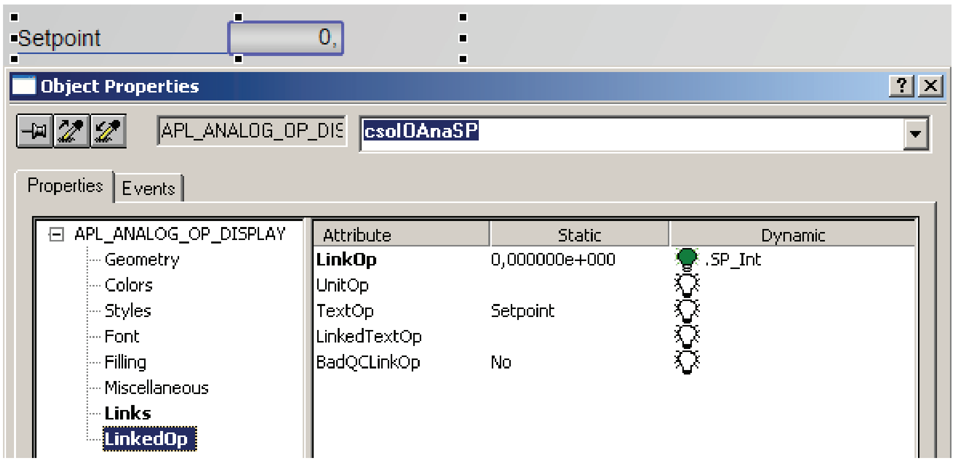
You set the user-specific authorization level at the "OperationLevel" attribute.

Authorization levels of the objects

- 1 = Process operation
- 2 = Higher order process operation
- 3 = Highest order process operation

Operating area

- The type of operating area is selected by means of the"OpAreaPicture" attribute. For normal analog value operation, there is the choice between
  - @PG_APL_OA_Analog100 -> without measuring range
  - @PG_APL_OA_Analog101 -> with measuring range
  - @PG_APL_OA_AnalogWithLimits -> with measuring range, slider, incremental operation.
- You determine the behavior after execution of the operation with the "OkOption" attribute. In the case of simple analog value operation this is always 3.
- In the "LinkedOp" tab (shown in the following figure) only the "LinkOp" property is of importance, the remaining properties are reserves.



Attribute	Static	Dynamic
LinkOp	0,000000e+000	.SP_Int
UnitOp		
TextOp	Setpoint	
LinkedTextOp		
BadQCLinkOp	No	

See also

General specifications in the screen @PG_MyAPLFP_Standard.pdl (Page 90)

Arranging display and operating objects (Page 21)

Configuring the analog value display (Page 21)

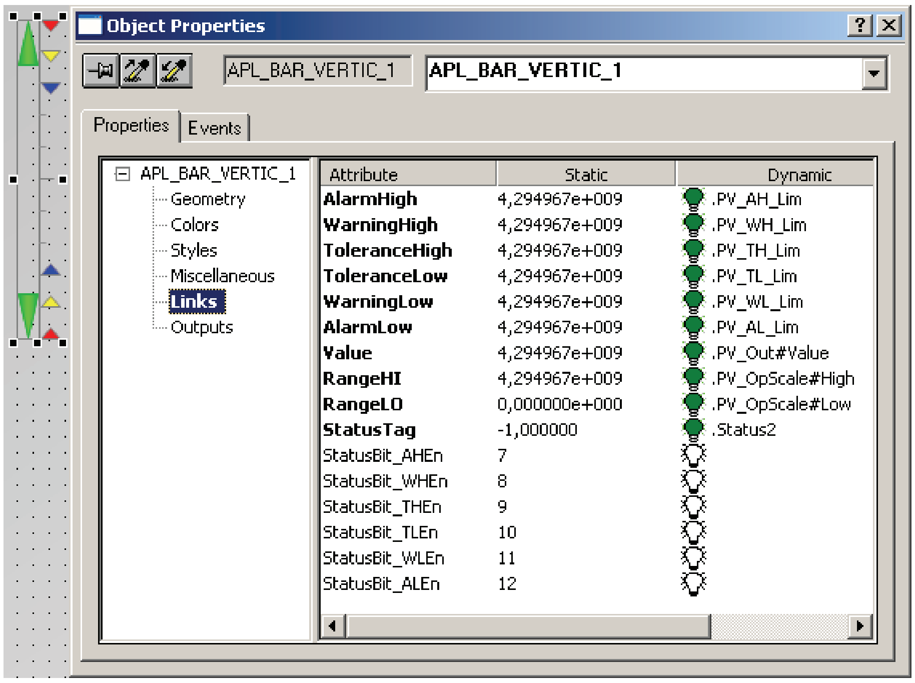
6.4.5.3 Bar graphs

The following section contains the properties of the bar graphs used in the APL faceplates that you set with parameters:

- Vertical bar graph for actual value with up to three limit value pairs
- Vertical bar graph for setpoint with a limit value pair
- Horizontal bar graphs

Vertical bar graph for actual value with up to three limit value pairs

1. Copy the "APL_BAR_VERTIC_1"(@PCS7ElementsAPL.pdl; Bar graph display) object from the template screen and insert the object into your faceplate view.
2. Assign a meaningful object name.

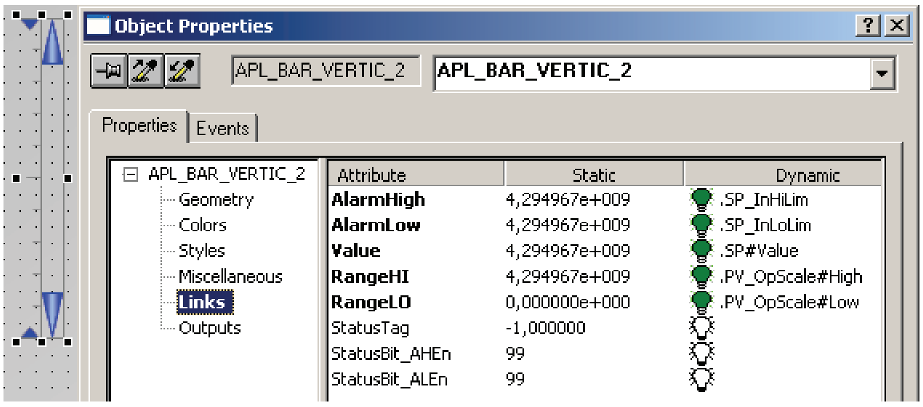


Attribute	Static	Dynamic
AlarmHigh	4,294967e+009	.PV_AH_Lim
WarningHigh	4,294967e+009	.PV_WH_Lim
ToleranceHigh	4,294967e+009	.PV_TH_Lim
ToleranceLow	4,294967e+009	.PV_TL_Lim
WarningLow	4,294967e+009	.PV_WL_Lim
AlarmLow	4,294967e+009	.PV_AL_Lim
Value	4,294967e+009	.PV_Out#Value
RangeHI	4,294967e+009	.PV_OpScale#High
RangeLO	0,000000e+000	.PV_OpScale#Low
StatusTag	-1,000000	.Status2
StatusBit_AHEn	7	
StatusBit_WHEn	8	
StatusBit_THEn	9	
StatusBit_TLEn	10	
StatusBit_WLEn	11	
StatusBit_ALEn	12	

Vertical bar graph for setpoint with a limit value pair

1. Copy the "APL_BAR_VERTIC_2"(@PCS7ElementsAPL.pdl; Bar graph display) object from the template screen and insert the object into your faceplate view.

2. Assign a meaningful object name.



Attribute	Static	Dynamic
AlarmHigh	4,294967e+009	.SP_InHiLim
AlarmLow	4,294967e+009	.SP_InLoLim
Value	4,294967e+009	.SP#Value
RangeHI	4,294967e+009	.PV_OpScale#High
RangeLO	0,000000e+000	.PV_OpScale#Low
StatusTag	-1,000000	
StatusBit_AHEn	99	
StatusBit_ALEn	99	

In principle the same procedure applies here as for the APL_BAR_VERTIC_1 bar graph.

Differences here are the blue bar color and the display of only one limit value pair that cannot be displayed / hidden event-controlled.

Using properties for displaying / hiding the limit value pairs

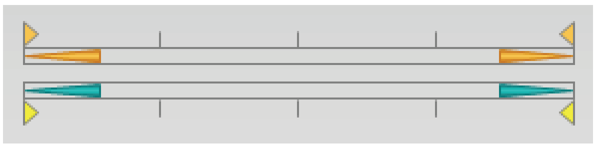
The following is required if you want to display or hide the limit value pair:

1. Apply a status word to the "StatusTag" property.
2. Create a script at the "StatusTag" property of the event.

You can use the corresponding script of the APL_BAR_VERTIC_1(@PCS7ElementsAPL.pdl; Bar graph display) object as the copy template.

Horizontal bar graphs

1. Copy the APL_BAR_HORIZ_1 or APL_BAR_HORIZ_2 (@PCS7ElementsAPL.pdl; Bar graph display) object from the template screen and insert the object into your faceplate view.
2. Assign a meaningful object name.



The horizontal bar graphs are identical with the APL_BAR_VERTIC_1 bar graph apart from the color and the horizontal position.

Attributes of the bar graph

Attribute	Meaning	Setting
Bar graph	"Value" property	Use the "Value" property to interconnect the analog value you want to display in the bar.
	"Colors" dialog	The color representation is specified in the "Colors" dialog. Standard color of the bar: Green
Specifying the measuring value range	"RangeHI" and "RangeLO" properties	Use the "RangeHI" and "RangeLO" properties to interconnect the measured value you want to display.
Limit values	Properties for limit values	You can have limit values displayed at the bar with the following properties: • "AlarmHigh" • "WarningHigh" • "ToleranceHigh" • "AlarmLow" • "Warninglow" • "ToleranceLow" The properties are represented as follows: • Red arrows for alarm • Yellow arrows for warning • Blue arrows for tolerance
"StatusTag" property	"StatusTag" property	Use the "StatusTag" property to interconnect a status word. • The underlying properties "StatusBit_xxxx" specify a corresponding bit position in the status word. Example: "StatusBit_0" for Bit 0 • Whether a limit value mark is displayed or not is specified event-controlled. Example: The status word "Status2" Bit 7 at the AS block is 1. The limit value mark "Alarm High" is then displayed at the bar.

See also

Configuring a bar graph (Page 27)

6.4.6 Displaying and operating binary values

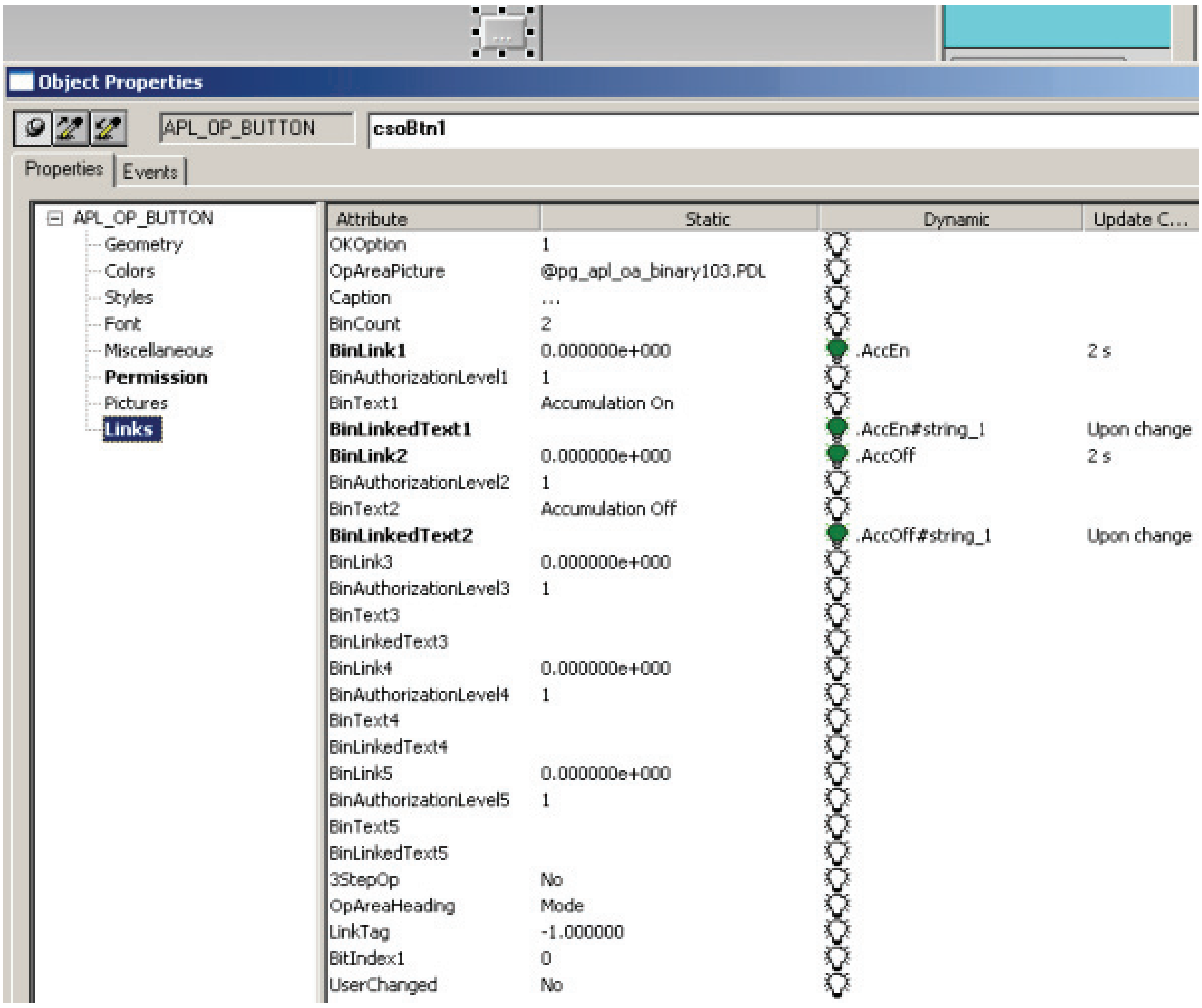
6.4.6.1 Binary value operation APL_OP_BUTTON

The following section describes the parameter assignment for a binary value operation.

Procedure

1. Copy the **APL_OP_BUTTON** (@PCS7ElementsAPL.pdl; Open binary operation area; Schaltfläche unter "Normal") object from the template screen and insert the object into your faceplate view.
   For information on positioning please refer to " ShowSelectionBorder (Page 98)".
2. Assign a meaningful object name, such as "csoBtnAccEn".
3. Enter the object name in the ""stUser"" object of your faceplate to update the operator authorization when the user changes.

For additional information refer to the chapter "General specifications in the screen @PG_MyAPLFP_Standard.pdl (Page 90)".



Attribute	Static	Dynamic	Update C...
OKOption	1		
OpAreaPicture	@pg_apl_oa_binary103.PDL		
Caption	...		
BinCount	2		
BinLink1	0.000000e+000	.AccEn	2 s
BinAuthorizationLevel1	1		
BinText1	Accumulation On		
BinLinkedText1		.AccEn#string_1	Upon change
BinLink2	0.000000e+000	.AccOff	2 s
BinAuthorizationLevel2	1		
BinText2	Accumulation Off		
BinLinkedText2		.AccOff#string_1	Upon change
BinLink3	0.000000e+000		
BinAuthorizationLevel3	1		
BinText3			
BinLinkedText3			
BinLink4	0.000000e+000		
BinAuthorizationLevel4	1		
BinText4			
BinLinkedText4			
BinLink5	0.000000e+000		
BinAuthorizationLevel5	1		
BinText5			
BinLinkedText5			
3StepOp	No		
OpAreaHeading	Mode		
LinkTag	-1.000000		
BitIndex1	0		
UserChanged	No		

Brief description of the properties relevant for binary value operation.

Detailed description will follow.

Property	Brief description
Links/ OkOption	Evaluation of the input value after execution OK
Links/ OpAreaPicture	Selection of the operating area (for example, one row or two rows)
Links/ Caption	Labeling of the selection button
Links/ BinCount	Number of the binary elements to be operated
Links/ BinLink1	Interconnection to the binary element 1
Links/ BinAuthorizationLevel1	User authorization for the binary element 1
Links/ BinText1	Fixed text in the faceplate for binary element 1
Links/ BinLinkedText1	Instance-specific text using S7_string for binary element 1
Links/ OpAreaHeading	Labeling for operating area
Permission/ PermissionTag	Variable of the process-related enable (OS_PERM)
Permission/ BinPermBitIndex1	Bit address in the word for the process-related operator authorization for binary element 1

The APL_OP_BUTTON object is designed so that you can create an operating dialog with up to five buttons.

OkOption

The following items are specified with the "OkOption" property:

- The written variable type
- The behavior after execution of the operation

Parameter assignment

- OkOption = 1

  When the corresponding button is selected, a logical "1" is written to the binary variables that are connected under BinLink1 to the maximum BinLink5.
- OkOption = 2
  Usage for the binary variables that have the logical values "0" and "1" written to them. In this case the same variable is interconnected to BinLink1 and BinLink2.
  - A logic "1" is written to the variable connected under BinLink1.
  - A logic "0" is written to the variable connected under BinLink2.
  - BinLink3 to 5 do not have any meaning in this case.
- OkOption = 4

  Only with the object APL_OP_BUTTON3
  - An integer variable is interconnected to BinLink1.
  - The value that is configured in the "AnalogValue" property is written to the integer variable interconnected to BinLink1.
- OkOption = 5

  The inverted value is written back to the binary value connected under BinLink1 (changeover function).

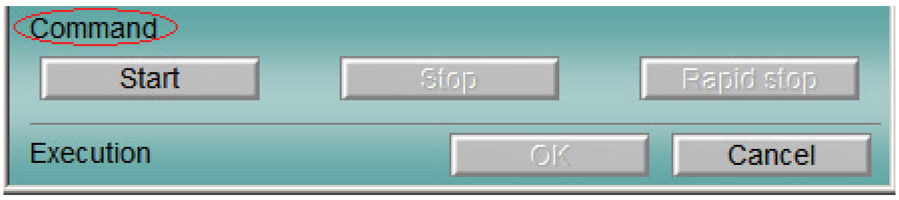
OpAreaPicture

At this property you configure an operating area with three or five buttons.

- "@pg_apl_oa_binary103.PDL" Operating area with up to three buttons
- "@pg_apl_oa_binary105.PDL" Operating area with up to five buttons

Header of the operating area

You configure the header of the operating area at the "OpAreaHeading" property.



BinCount

Here you configure the number of buttons and used variable connections to BinLink1 to BinLink5.

BinAuthorizationLevel1 .. 5

Use "BinAuthorizationLevel1 .. 5" to set an individual operator authorization level for the individual operations.

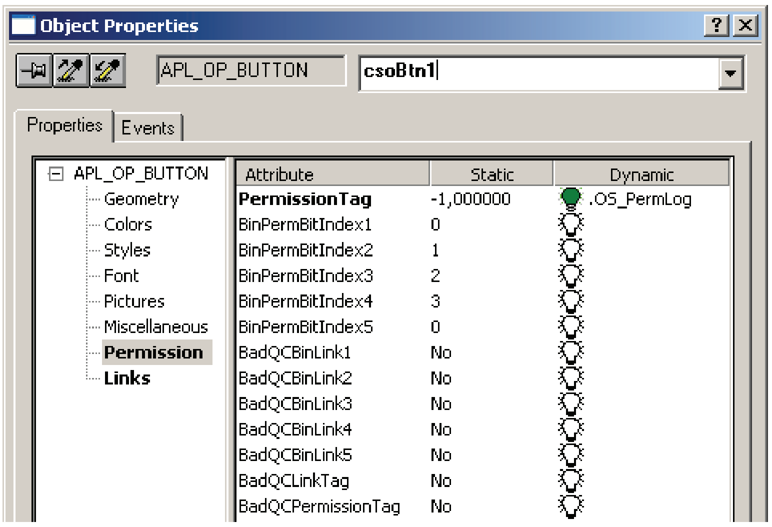
BinText1 .. 5

Use "BinText1 .. 5" to configure the button labeling.

BinLinkedText1 .. 5

If you interconnect the corresponding internal OS variable to "BinLinkedText1 .. 5" , the button labeling can be used by the S7_String0/1 configured at the block.

If no text is entered in S7_string, the text that is entered under BinText1 .. 5 will be used.



Attribute	Static	Dynamic
PermissionTag	-1,000000	.OS_PermLog
BinPermBitIndex1	0	
BinPermBitIndex2	1	
BinPermBitIndex3	2	
BinPermBitIndex4	3	
BinPermBitIndex5	0	
BadQCBinLink1	No	
BadQCBinLink2	No	
BadQCBinLink3	No	
BadQCBinLink4	No	
BadQCBinLink5	No	
BadQCLinkTag	No	
BadQCPermissionTag	No	

PermissionTag

The process-related enable is connected to this property. The double word "OS_PermLog" is usually interconnected here.

Use "BinPermBitIndex1 .. 5" to assign parameters to the corresponding bit (starting from Bit0) in the double word for the individual operations.

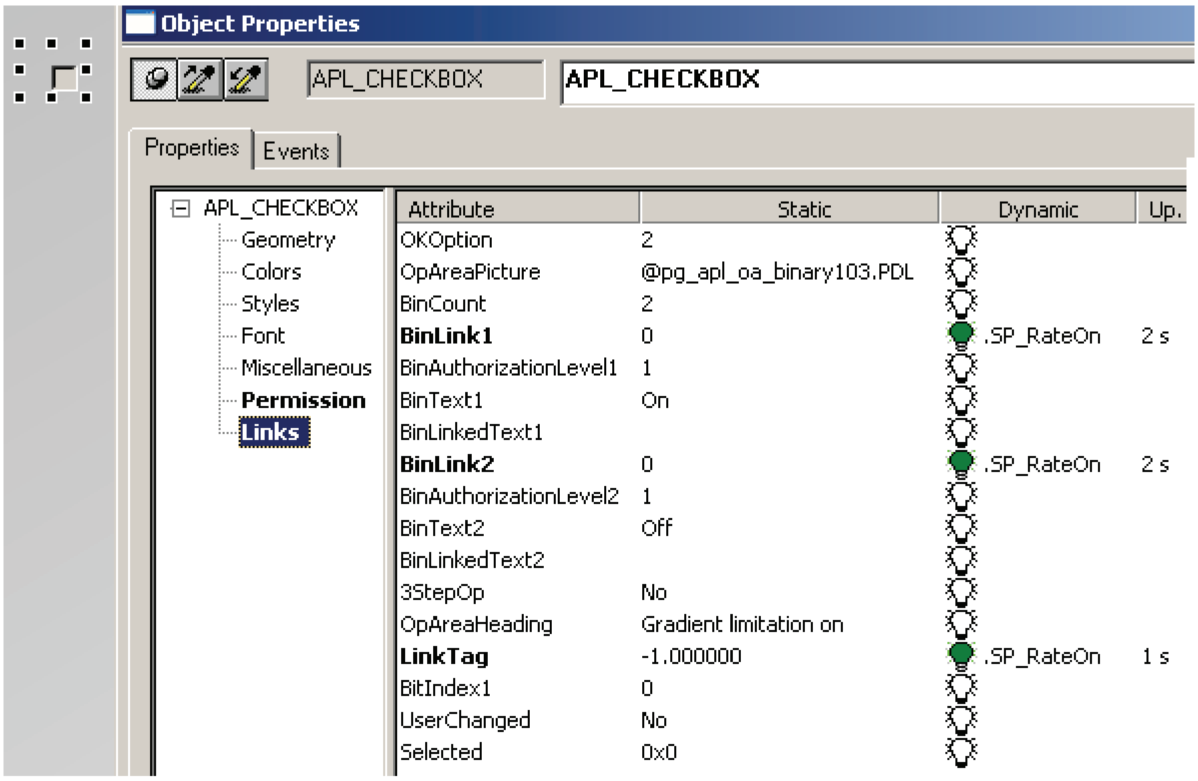
6.4.6.2 Check box

The following section describes the inclusion of a check box.

Procedure

1. Copy the APL_CHECKBOX (@PCS7ElementsAPL.pdl; Check box) object from the template screen and insert the object into your faceplate view.
   For information on positioning please refer to " ShowSelectionBorder (Page 98)".
2. Assign a meaningful object name, such as "APL_CHECKBOX_Reset".
3. In the "stUser" object in the screen @PG_MyAPLFP_Standard.pdl enter the object name for an update of the operator authorization when the user changes.

For additional information refer to the chapter "General specifications in the screen @PG_MyAPLFP_Standard.pdl (Page 90)".



Attribute	Static	Dynamic	Up.
OKOption	2		
OpAreaPicture	@pg_apl_oa_binary103.PDL		
BinCount	2		
BinLink1	0	.SP_RateOn	2 s
BinAuthorizationLevel1	1		
BinText1	On		
BinLinkedText1			
BinLink2	0	.SP_RateOn	2 s
BinAuthorizationLevel2	1		
BinText2	Off		
BinLinkedText2			
3StepOp	No		
OpAreaHeading	Gradient limitation on		
LinkTag	-1.000000	.SP_RateOn	1 s
BitIndex1	0		
UserChanged	No		
Selected	0x0		

Figure 6-6 CheckBox1

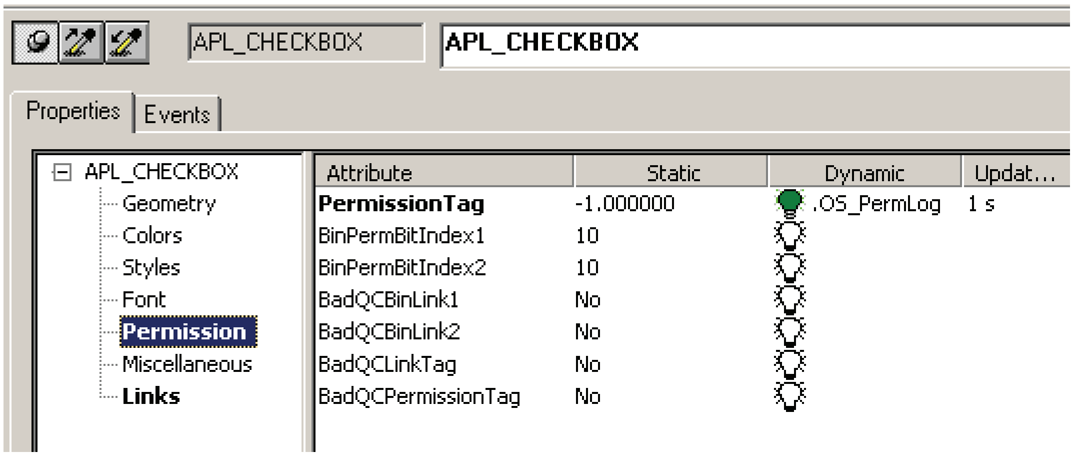
Brief description of the properties relevant for the check box.

Detailed description will follow.

Property	Brief description
Links/ OkOption	Evaluation of the input value after execution OK (1 or 2)
Links/ OpAreaPicture	Selection of the operating area (for example, one row or two rows)
Links/ BinCount	Number of elements to be operated (always 2 here)
Links/ BinLink1	Interconnection to the binary element 1
Links/ BinAuthorizationLevel1	User authorization for the binary element 1
Links/ BinText1 Fester	Text in the faceplate for binary element 1
Links/ BinLinkedText1	Instance-specific text using S7_string for binary element 1
Links/ OpAreaHeading	Labeling for operating area
Links/ LinkTag	Variable for display of the check mark in the check box
Links/ BitIndex1	Bit position when there is a status word at the LinkTag
Permission/ PermissionTag	Variable of the process-related enable (OS_PERM)
Permission/ BinPermBitIndex1	Bit address in the word for the process-related operator authorization for binary element 1

Properties

Property	Description
OKOption	You can use the APL check box with the "OKOption" property. The properties "BinLink1" and "BinLink2" are evaluated through the usage of the "OKOption" property. The following settings are possible: • Property "OKOption" =1 Two different variables are interconnected to the properties "BinLink1" and "BinLink2" . A logical 1 is written to these variables. • Property "OKOption" =2 The same variable (for example SP_RateOn, see CheckBox1 graphics) is interconnected to the properties "BinLink1" and "BinLink2" . – BinLink1 stands for the setting to logical 1 of the boolean variable – BinLink0 stands for the setting to logical 0 of the boolean variable
BinAuthorizationLevel1 **BinAuthorizationLevel2**	You can set an individual operator authorization level for operations by using the "BinAuthorizationLevel1 ..2" properties.
LinkTag	Use the "LinkTag" property to interconnect the variable for displaying the check mark in the check box, logical 1 = check mark set.
BitIndex1	Use the "BitIndex1" property to parameterize the corresponding bit (starting from Bit0) in the status word, if you have interconnected a status word to LinkTag.
BinText1 BinText2	Use the BinText1 .. 2" properties to configure the button labeling.
BinLinkedText1 BinLinkedText2	• If you interconnect the corresponding OS-internal variable to "BinLinkedText1 .. 2" , the button labeling can come from the S7_string0/1 configured at the block. • If no text is entered at the block for "S7_string0/1" , the text that is entered at the BinText1 .. 2" properties will be used.



Attribute	Static	Dynamic	Updat...
PermissionTag	-1.000000	.OS_PermLog	1 s
BinPermBitIndex1	10		
BinPermBitIndex2	10		
BadQCBinLink1	No		
BadQCBinLink2	No		
BadQCLinkTag	No		
BadQCPermissionTag	No		

PermissionTag

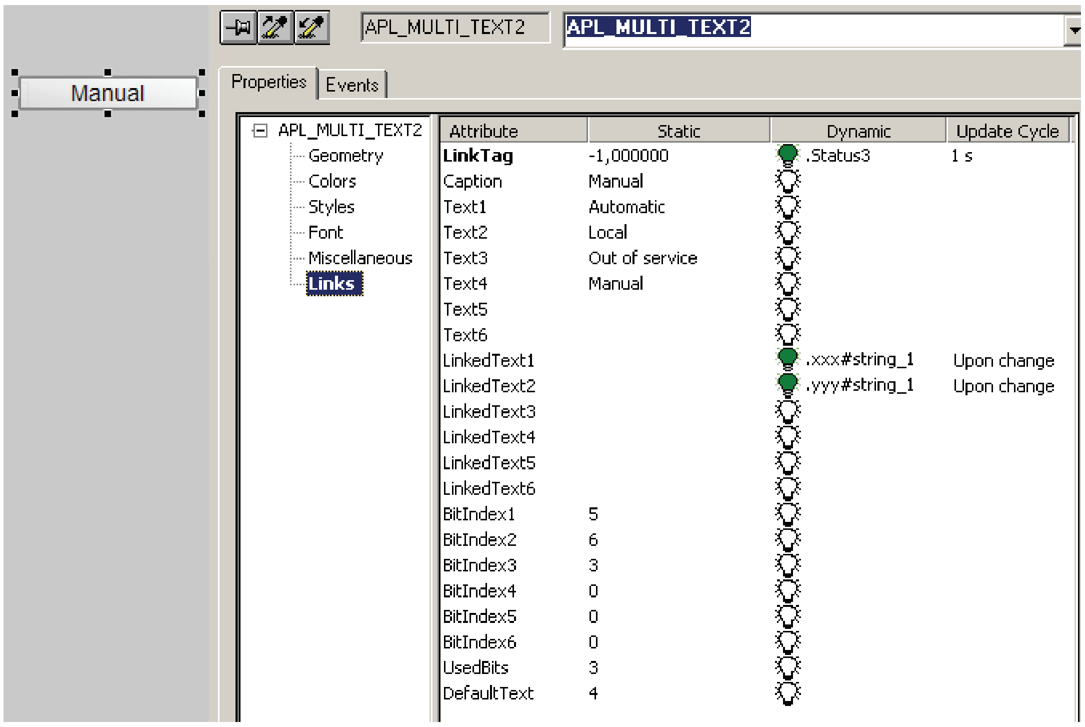
The "PermissionTag" property is used to interconnect the process-related enable.

- Recommendation:
  Use the double word "OS_PermLog".
  Use "BinPermBitIndex1 .. 2" to assign parameters to the bit in the double word (starting from Bit0) that contains the enable.
  - BinPermIndex1 is the operator authorization for logic 1 setting of the variable.
  - BinPermIndex2 is the operator authorization for logic 0 setting of the variable.
- The following is required if you do not want to connect a variable to the "PermissionTag" property:
  - Parameterize the value in "PermissionTag" from -1 to +1.
  - Address Bit0 at "BinPermBitIndex1 .. 2"

6.4.6.3 Binary display with text boxes

Procedure

1. Copy the **APL_MULTI_TEXT2** (@PCS7ElementsAPL.pdl; Status display text; Manual) object from the template screen and insert the object into your faceplate view.
2. Assign a meaningful object name.



Attribute	Static	Dynamic	Update Cycle
LinkTag	-1,000000	.Status3	1 s
Caption	Manual		
Text1	Automatic		
Text2	Local		
Text3	Out of service		
Text4	Manual		
Text5			
Text6			
LinkedText1		.xxx#string_1	Upon change
LinkedText2		.yyy#string_1	Upon change
LinkedText3			
LinkedText4			
LinkedText5			
LinkedText6			
BitIndex1	5		
BitIndex2	6		
BitIndex3	3		
BitIndex4	0		
BitIndex5	0		
BitIndex6	0		
UsedBits	3		
DefaultText	4		

Figure 6-7 MultiText2_1

Brief description of the properties relevant for the text display.

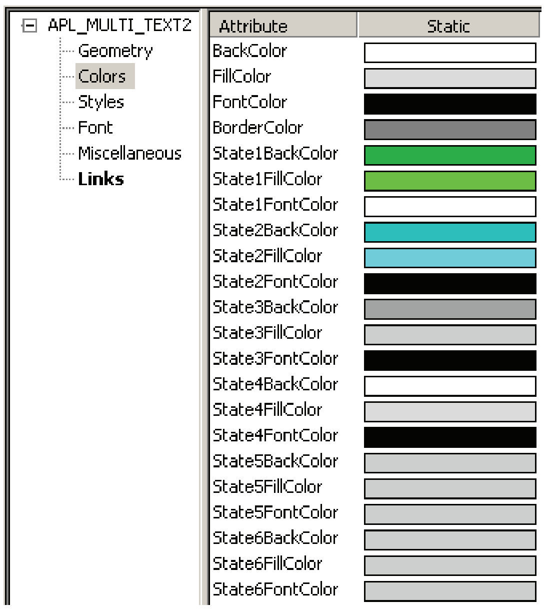
Detailed description will follow.

Property	Brief description
Links/ LinkTag	Applying a status word
Links/ Caption	Text actually displayed
Links/ Text1 .. 6	Parameterized text in the faceplate for the alternatives
Links/ LinkedText1..6	Text from the instance for the alternatives (only available with specific versions)
Links/ BitIndex1..6	Assignment of the bits in the status word for the alternatives
Links/ UsedBits	Number of used bits beginning with BitIndex1
Links/ DefaultText	Default when no bit is set
Colors/ State1BackColor	Background color Text1
Colors/ State1FillColor	Background fill pattern color for Text1
Colors/ State1FontColor	Text color for Text1

The object is basically designed so that you can have up to six different texts displayed in the text display depending on the bits of a status word.

Property	Brief description
LinkTag	The corresponding status word is interconnected to the **"LinkTag"** property.
BitIndex1 ...BitIndex6	You parameterize the corresponding bits (starting from Bit0) in the status word which display "Text1" to "Text6" at the **"BitIndex1 ..6"** properties.
LinkedText1... LinkedText6	If a string of the "string0/ string1" type at which a text is configured, is interconnected to the **"LinkedText1 ... 6"** properties, this instance-specific text will be displayed. Available at the objects APL_MULTI_TEXT6 and APL_MULTI_TEXT.
UsedBits	The **"UsedBits""** property is used to specify the number of alternatives of the text display. Starting from BitIndex1 you determine the bits used in the status word.
DefaultText	Use the property **"DefaultText"** to determine the text that will be displayed when none of the addressed bits = 1. Permitted values are 1 to 6. **Example:** "DefaultText" = 4 and "UsedBits" = 3 If none of the bits addressed under BitIndex1 .. 3 is 1, then the text configured under Text4 will be displayed.

Configure the colors for the individual texts in the "Colors" tab.



- APL_MULTI_TEXT2
  - Geometry
  - Colors
  - Styles
  - Font
  - Miscellaneous
  - **Links**

Attribute	Static
BackColor	⬜
FillColor	▨
FontColor	⬛
BorderColor	▨
State1BackColor	🟩
State1FillColor	🟩
State1FontColor	⬜
State2BackColor	🟦
State2FillColor	🟦
State2FontColor	⬛
State3BackColor	▨
State3FillColor	▨
State3FontColor	⬛
State4BackColor	⬜
State4FillColor	▨
State4FontColor	⬛
State5BackColor	▨
State5FillColor	▨
State5FontColor	▨
State6BackColor	▨
State6FillColor	▨
State6FontColor	▨

Additional text displays

- APL_MULTI_TEXT6

  Interconnected variables can be displayed. If necessary, the font size of the displayed texts will be adapted. If the interconnected variables are empty, then the standard texts will be used.

- APL_MULTI_TEXT

  Like APL_MULTI_TEXT6, in addition, an external object is adapted to the displayed text in its position.

  An external object could be a status display, for example, that is to be displayed in addition to the text.

  You configure the object names of the external object in the "ArrowDisplay" tab of the "ArrowObjectName" property.

  – Take into account that if this object is used, the configured external object must exist, because errors in the processing of the script may otherwise occur.

  – At interconnected texts the external object is switched to invisible.

- APL_MULTI_TEXT4

  This display is transparent, meaning that it is also possible to display no texts.

- APL_MULTI_TEXT3

  This display is transparent, meaning that it is also possible to display no texts. The different versions are all displayed in the same color, which means, runtime optimized.

See also

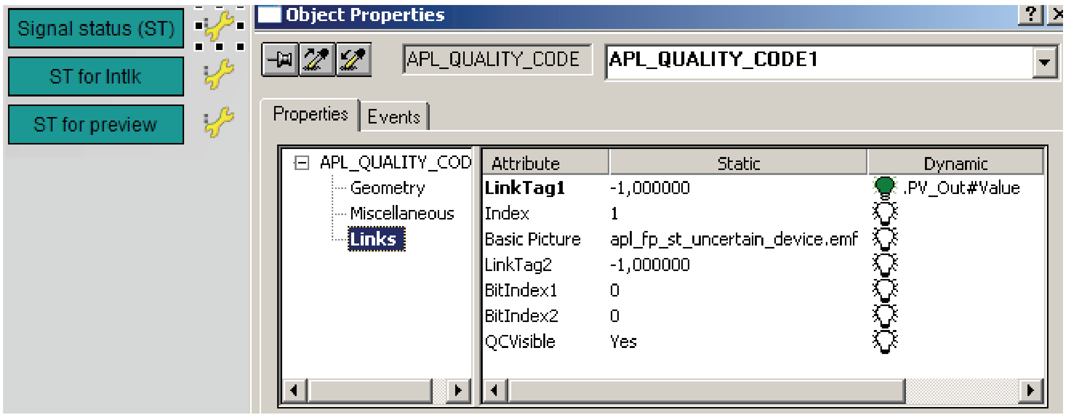
Configuring the display and changeover of the operating mode (Page 25)

6.4.6.4 Signal status display

The following section describes the display of the signal status:

Procedure

1. Copy the APL_QUALITY_CODE<x>(@PCS7ElementsAPL.pdl; Symbol status display) object from the template screen. Insert the object into your faceplate view.
2. Assign a meaningful object name.



APL_QUALITY_CODE APL_QUALITY_CODE1

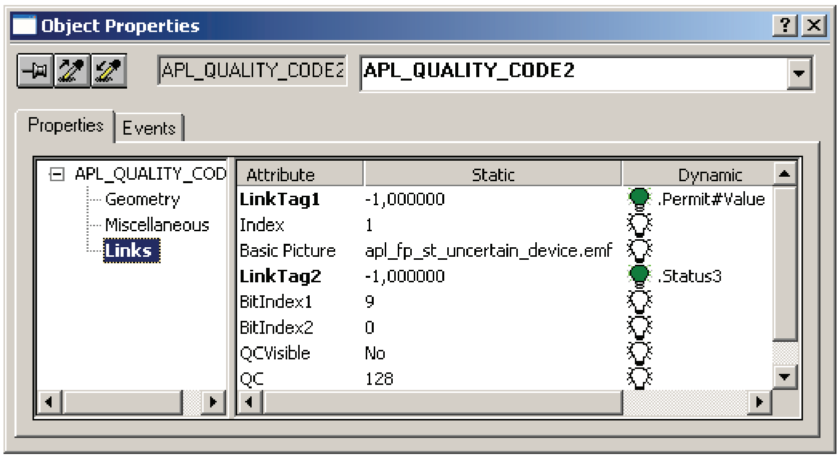
Attribute	Static	Dynamic
LinkTag1	-1,000000	.PV_Out#Value
Index	1	
Basic Picture	apl_fp_st_uncertain_device.emf	
LinkTag2	-1,000000	
BitIndex1	0	
BitIndex2	0	
QCVisible	Yes	

3. Interconnect the variable whose signal status you wish to display with the "LinkTag1" property and place your object in your view.

Faceplates in the template screen

The template screen contains the following faceplates for displaying the signal status:

- APL_QUALITY_CODE1
  Interconnect the variable whose Signal Status you wish to display with the "LinkTag1" property and place your object in your view.
- APL_QUALITY_CODE2
  This display for a Signal Status is intended primarily for interlock blocks because here a Signal Status 16#60 is displayed as a white **B** on a blue background.
  Interconnect the variable whose Signal Status you wish to display with the "LinkTag1" property and place your object in your view.
  In addition you can also specify with this version if the symbol is generally displayed or not.
  Application example:
  If an input is not interconnected, nothing should be displayed.
  – The "LinkTag2" property is used to interconnect a status word.
  – Use "BitIndex1" to address the corresponding bit in the status word.



Attribute	Static	Dynamic
LinkTag1	-1,000000	.Permit#Value
Index	1	
Basic Picture	apl_fp_st_uncertain_device.emf	
LinkTag2	-1,000000	.Status3
BitIndex1	9	
BitIndex2	0	
QCVisible	No	
QC	128	

- APL_QUALITY_CODE3
  Like the APL_QUALITY_CODE2 display, but a yellow hand is displayed at Signal Status 16#60 .

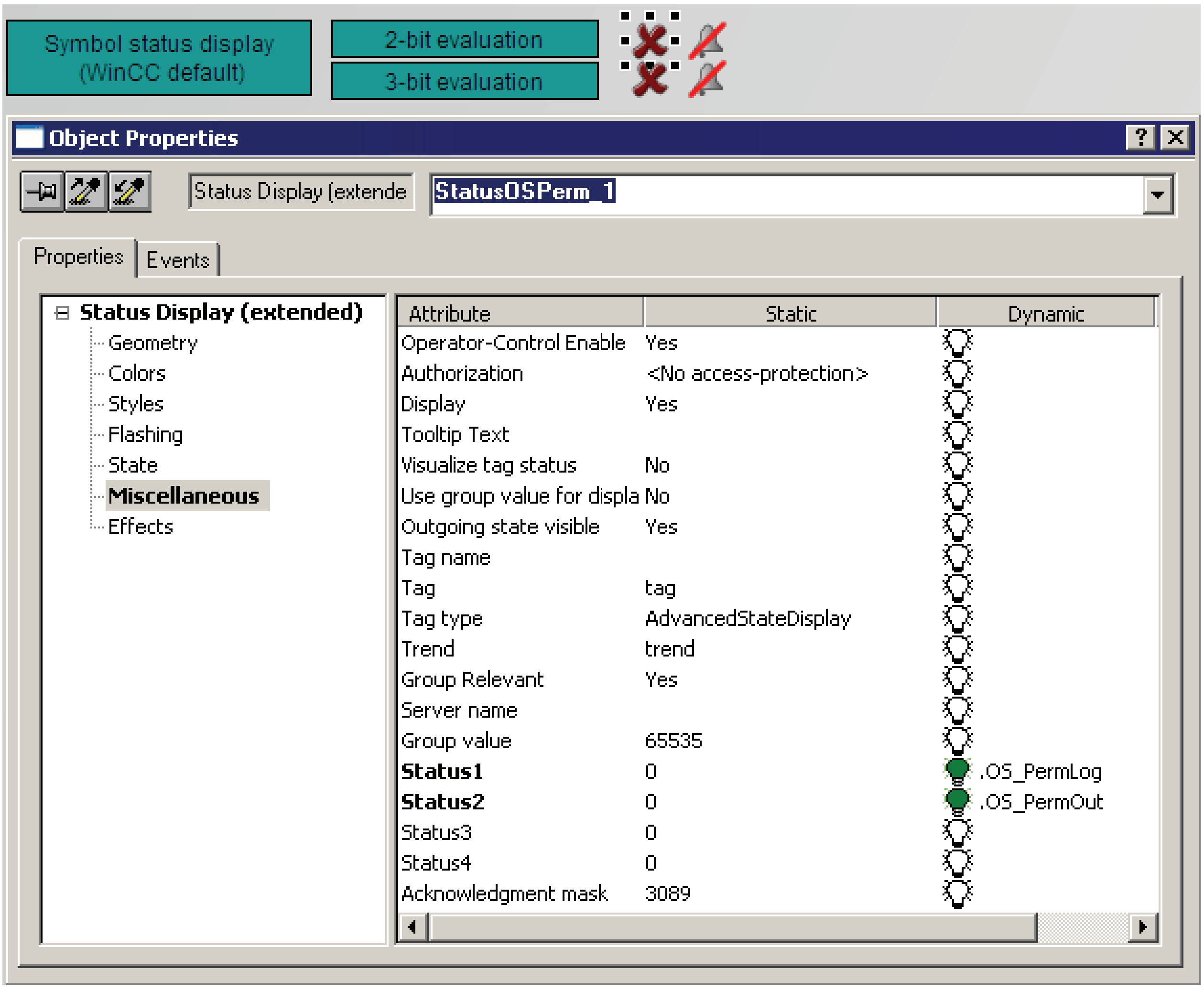
6.4.6.5 Displaying the OS_Perm operating locks

Use this object to display the state of the operator control enable of an object:

- Green check mark -> Operation enabled by the AS block.
- Gray check mark -> Operation is not possible due to the process
- Red cross -> Operation disabled by the AS block.

Procedure

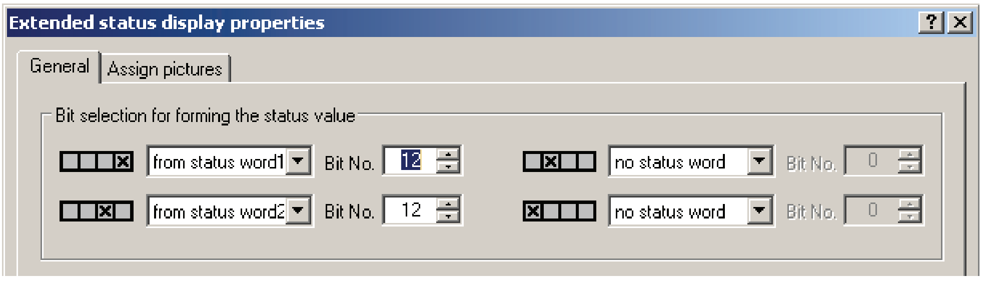
1. Copy the "StatusOSPerm_1" object from the template screen. Insert the object into your faceplate view.
   The object is an extended status display.
2. Assign a meaningful object name.



Attribute	Static	Dynamic
Operator-Control Enable	Yes	
Authorization	<No access-protection>	
Display	Yes	
Tooltip Text		
Visualize tag status	No	
Use group value for displa	No	
Outgoing state visible	Yes	
Tag name		
Tag	tag	
Tag type	AdvancedStateDisplay	
Trend	trend	
Group Relevant	Yes	
Server name		
Group value	65535	
Status1	0	.OS_PermLog
Status2	0	.OS_PermOut
Status3	0	
Status4	0	
Acknowledgment mask	3089	

Figure 6-8 OS_Perm5

3. Interconnect the following properties with parameters (see figure "OS_Perm5"):
   – "Status1" property with "OS_PermLog" parameter
   – "Status2" property with "OS_PermOut" parameter
4. In the configuration dialog of the extended status display assign parameters to the corresponding bit of "OS_PermLog" and "OS_PermOut" (see figure "OS_Perm6").



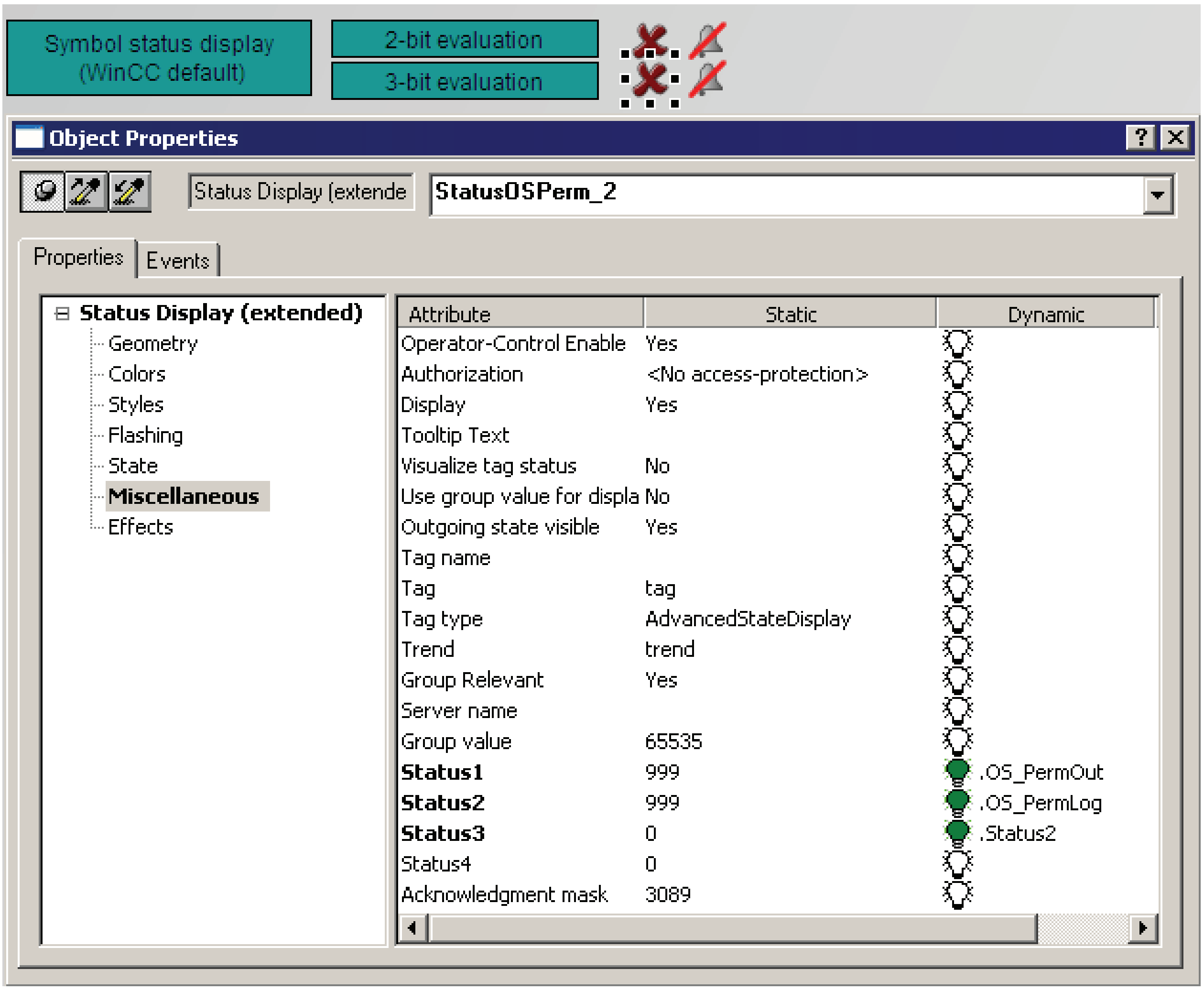
Bit selection for forming the status value

☐☐☐☒	from status word1	Bit No.	12	☐☒☐☐	no status word	Bit No.	0
☐☐☒☐	from status word2	Bit No.	12	☒☐☐☐	no status word	Bit No.	0

Figure 6-9 OS_Perm6

Example "OS_Perm7"

The complete display can be switched to invisible in the display of the "StatusOSPerm_2" object (see figure "OS_Perm7") . To this purpose the corresponding bit in the status word is interconnected to "Status3" and is configured in the configuration dialog of the extended status display.



StatusOSPerm_2

Attribute	Static	Dynamic
Operator-Control Enable	Yes	
Authorization	<No access-protection>	
Display	Yes	
Tooltip Text		
Visualize tag status	No	
Use group value for displa	No	
Outgoing state visible	Yes	
Tag name		
Tag	tag	
Tag type	AdvancedStateDisplay	
Trend	trend	
Group Relevant	Yes	
Server name		
Group value	65535	
Status1	999	.OS_PermOut
Status2	999	.OS_PermLog
Status3	0	.Status2
Status4	0	
Acknowledgment mask	3089	

Figure 6-10 OS_Perm7

6.4.7 Creating your own bitmaps and status displays

The bitmaps or EMF files of the APL faceplates and block icons have been created with Visio 2007 and are stored as a source in Visio VSD format.

Files and formats

The VSD files are stored as "EMF+" files for use in PCS 7.

- This "EMF+" format offers the following:
  - Improved 3D views
  - Improved color gradients
  - Allows the size of the block icons to be adapted
- In Windows the "EMF+" files cannot be differentiated from EMF files (*.emf).
- Files that are stored as EMF files cannot be modified with Visio 2007 without the original VSD file.

6.4.8 Solutions for special requirements

6.4.8.1 Display of the block comment in the ToolTipText of the V7 block icons

By default, the tagname is displayed as a ToolTipText at APL block icons of version 7.

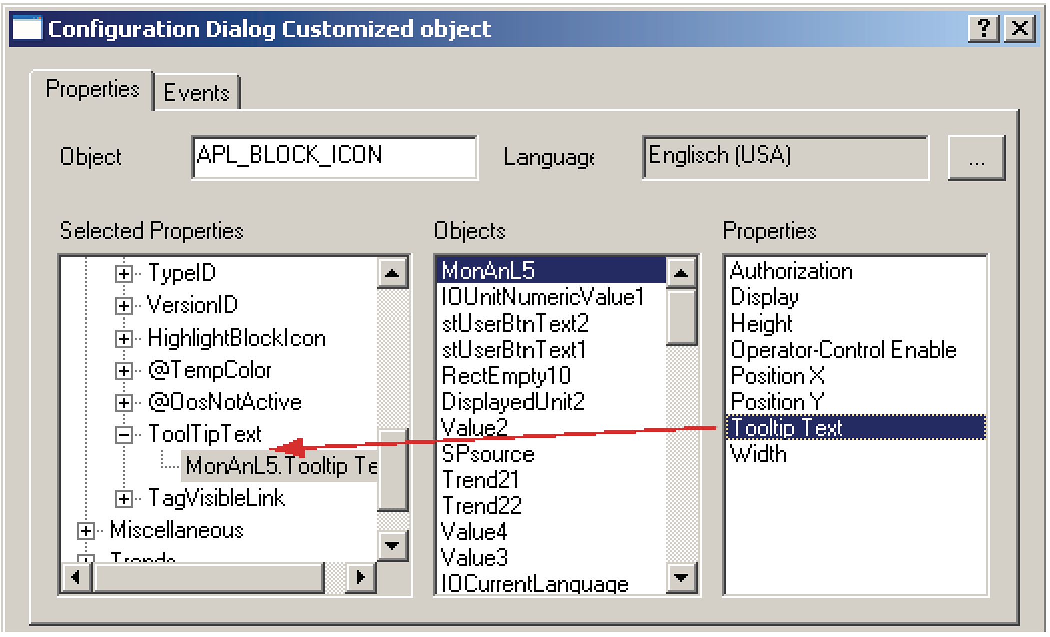
A block comment can be displayed as a TooltipText. The required changes are described below.

In the example, we use a copy of the APL block icon "MonAnl5" from the @PCS7TypicalsAPLV7.PDL (MonAnl - with Header) template screen.

Procedure

1. Select the block icon.
2. Use the shortcut menu (right-hand mouse button) to open the configuration dialog.
3. Select the "Systems/ToolTipText" property in the Selected Properties list.
4. Delete "IOTagname.OutputValue" in the subordinate object.
5. In the "Objects" list select the user object (in the "ToolTipText2" screen the "MonAnl5" object).

6. Drag the "Tooltip Text" entry from the "Properties" list onto the left-hand list "Selected Properties" "ToolTipText" property.
   The result is shown in the figure below.



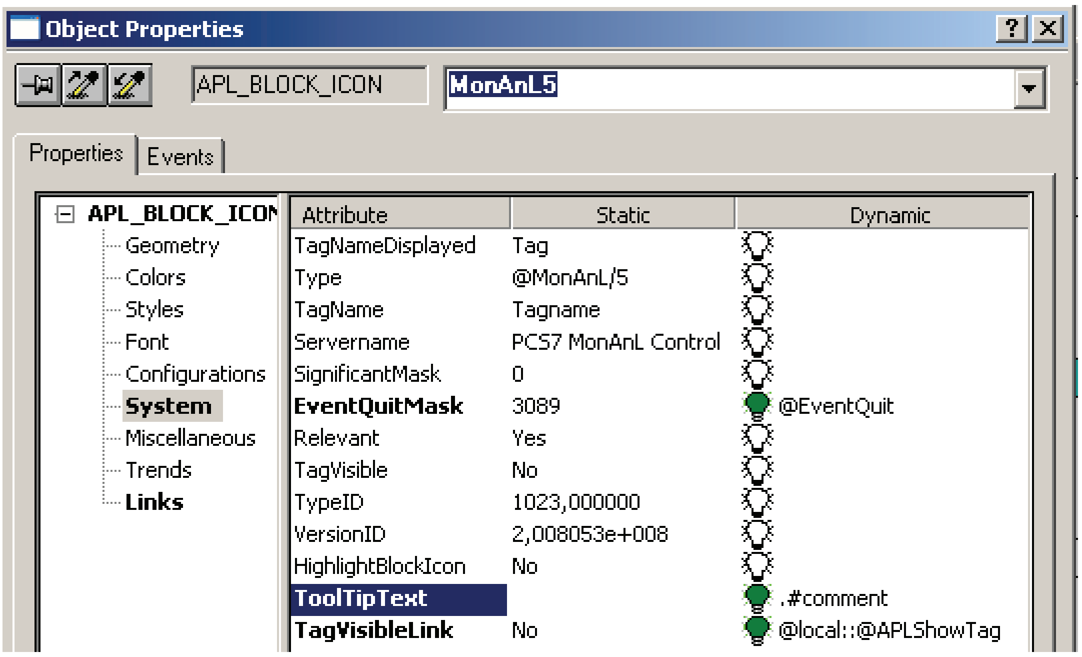
Configuration Dialog Customized object

Object: APL_BLOCK_ICON — Language: Englisch (USA)

Selected Properties	Objects	Properties
TypeID	MonAnL5	Authorization
VersionID	IOUnitNumericValue1	Display
HighlightBlockIcon	stUserBtnText2	Height
@TempColor	stUserBtnText1	Operator-Control Enable
@OosNotActive	RectEmpty10	Position X
ToolTipText	DisplayedUnit2	Position Y
MonAnL5.Tooltip Te	Value2	Tooltip Text
TagVisibleLink	SPsource	Width
Miscellaneous	Trend21	
	Trend22	
	Value4	
	Value3	
	IOCurrentLanguage	

Figure 6-11 ToolTipText2

7. Click "OK".
8. Configure the following in the properties of the block icon:
   The text ".#comment" in the "Systems" folder at the "ToolTipText" property for dynamics see figure "ToolTipText3"



Object Properties

APL_BLOCK_ICON — MonAnL5

APL_BLOCK_ICON: Geometry, Colors, Styles, Font, Configurations, **System**, Miscellaneous, Trends, **Links**

Attribute	Static	Dynamic
TagNameDisplayed	Tag	
Type	@MonAnL/5	
TagName	Tagname	
Servername	PCS7 MonAnL Control	
SignificantMask	0	
EventQuitMask	3089	@EventQuit
Relevant	Yes	
TagVisible	No	
TypeID	1023,000000	
VersionID	2,008053e+008	
HighlightBlockIcon	No	
ToolTipText		.#comment
TagVisibleLink	No	@local::@APLShowTag

Figure 6-12 ToolTipText3

6.4.8.2 Multi-instance faceplates

A multi-instance faceplate is a faceplate that you use to access different block instances in a CFC chart.

Procedure for multi-instance faceplates of the Standard Library

At the Standard Library the @Faceplate object has a MULTI_INSTANCE property that is set to TRUE. In this case the block name is then truncated at the OS tagname. In the multi-instance faceplate the corresponding block name then has to be supplemented at the parameter name of all the variable connections.

For example /Controller.PV_IN. instead of .PV_IN in the standard faceplate

This type of multi-instance technique is not supported for the APL.

Procedure for multi-instance faceplates of the APL

The following method is used for the APL faceplates. You can also use this method for the Standard Library.

Requirements

- The blocks you want to address are located in the same CFC chart.
- There is only one technological block with messages and messages do not have to be displayed for the other blocks.
- The block name of the other blocks contains the name of the technological block with messages + a name extension (see the example figure below "MultiInstanceFP_1").

Procedure

1. Create all blocks you want to address in the same CFC chart.
2. You form the name of the additional blocks from the name of the block with messages. This name contains a name extension.

Example

The example figure MultiInstanceFP_1 contains the following blocks:

- Block with messages:
  - Motor block, "MOT_".
- Blocks without messages:
  - Operating hours counter, "MOT_OH"
  - Switching cycle counter, "MOT_SC"

Displaying messages in the multi-instance faceplate

The multi-instance faceplate cannot display internal messages of the additional blocks. For the example this means that any messages existing in the operating hours counter and switching cycle counter are not displayed in the multi-instance faceplate.

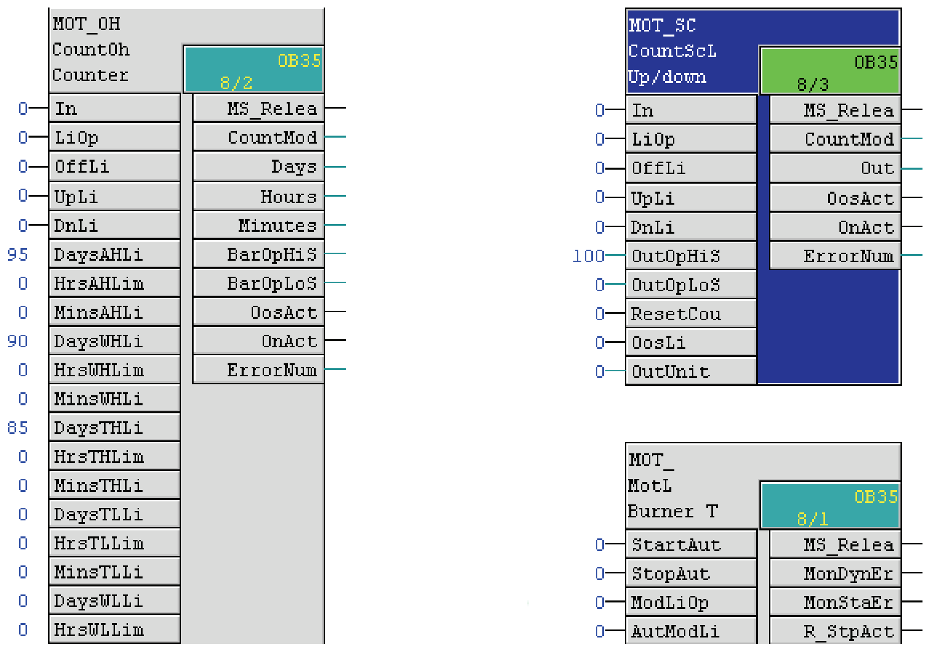


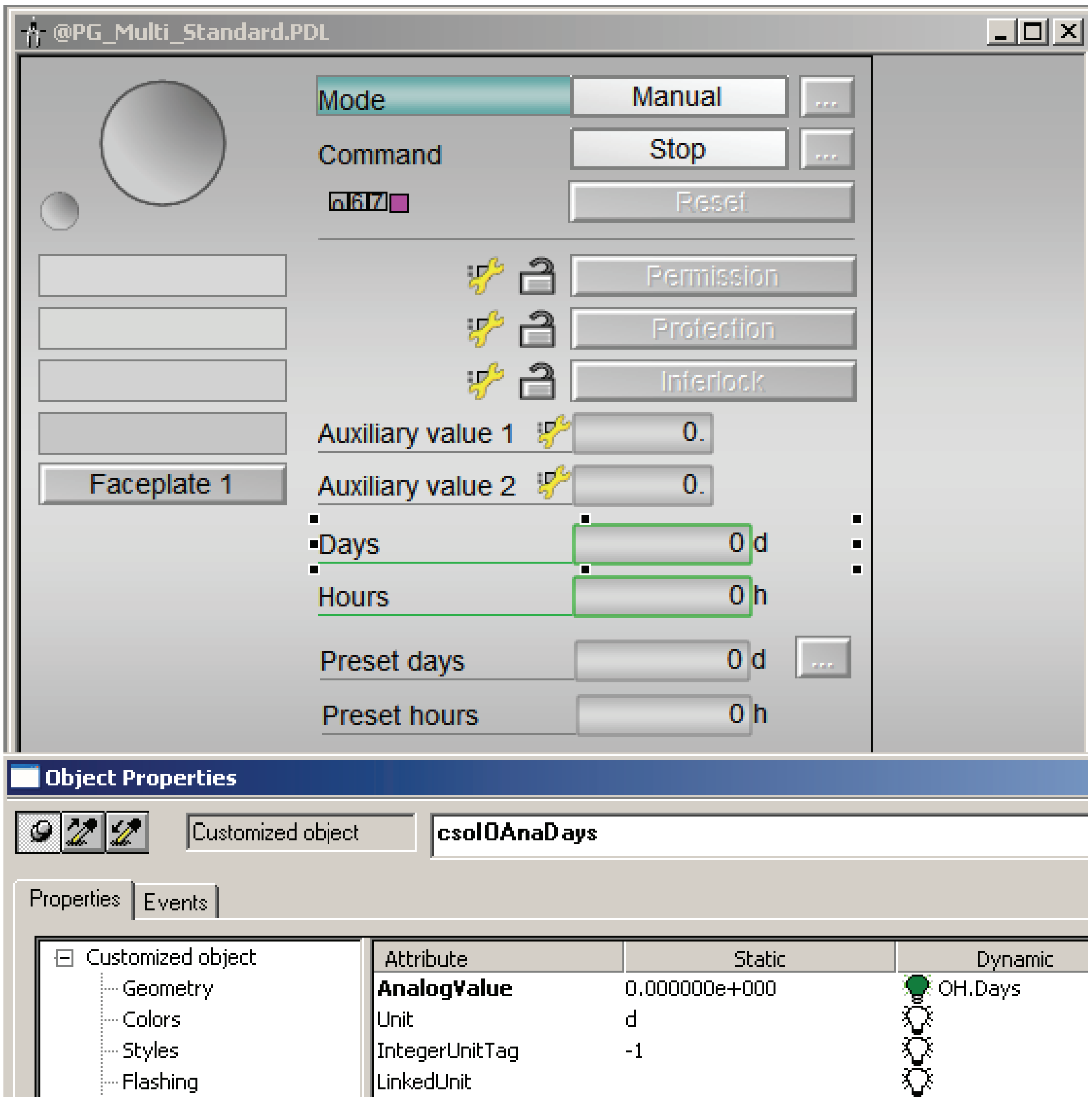
Figure 6-13 MultiInstanceFP_1

Connecting variables in the faceplate

- For the motor:
  No changes to the variable connections required.
- For the operating hours counter and the switching cycle counter:
  You have to carry out the corresponding name extensions in the faceplate at the variable connections.

Example

OH.days or SC.Out, see also MultiInstanceFP_2 graphics.



Auxiliary value 1 0.
Auxiliary value 2 0.
Days 0 d
Hours 0 h
Preset days 0 d
Preset hours 0 h

Customized object csol0AnaDays

Customized object
Geometry
Colors
Styles
Flashing

Attribute	Static	Dynamic
AnalogValue	0.000000e+000	OH.Days
Unit	d	
IntegerUnitTag	-1	
LinkedUnit		

Figure 6-14 MultiInstanceFP_2

Appendix A

A.1 Type ID of the object types

Overview

Object type	TypeID
@Faceplate	1
APL_ANA_EMPTY	21
APL_ANA_EMPTY2	26
APL_ANA_EMPTY3	25
APL_ANALOG_OP_DISPLAY	20
APL_ANALOG_OP_DISPLAY2	24
APL_ANALOG_OP_DISPLAY3	27
APL_ANALOG_OP_DISPLAY4	28
APL_BAR_GRADIENT_1	74
APL_BAR_HORIZ_1	72
APL_BAR_HORIZ_2	73
APL_BAR_LINE	84
APL_BAR_LINE_TEXT_E	82
APL_BAR_LINE_TEXT_R	83
APL_BAR_TEXT_E	80
APL_BAR_TEXT_R	81
APL_BAR_VERTIC_1	71
APL_BAR_VERTIC_2	70
APL_BAR_VERTIC_3	75
APL_BLOCK_ICON	3
APL_CHECKBOX	40
APL_FACEPLATE	50
APL_FACEPLATE2	51
APL_FACEPLATE3	52
APL_INTLK_DISPLAY	100
APL_INTLK_FIRSTIN	102
APL_INTLK_OUTPUT	103
APL_INTLK_OUTPUT2	104
APL_INTLK_STATUS	101
APL_LOCK_SYMBOL	60
APL_MULTI_TEXT	10
APL_MULTI_TEXT10	19

Object type	TypeID
APL_MULTI_TEXT11	120
APL_MULTI_TEXT2	11
APL_MULTI_TEXT3	12
APL_MULTI_TEXT4	13
APL_MULTI_TEXT5	14
APL_MULTI_TEXT6	15
APL_MULTI_TEXT7	16
APL_MULTI_TEXT8	17
APL_MULTI_TEXT9	18
APL_OBJ_COLLECTION	2
APL_OP_BUTTON	30
APL_OP_BUTTON2	31
APL_OP_BUTTON3	32
APL_OP_BUTTON4	33
APL_OP_BUTTON5	34
APL_OP_OBJECT	5
APL_OP_PERMISSION	6
APL_PIN_BUTTON	4
APL_QUALITY_CODE	61
APL_QUALITY_CODE2	62
APL_QUALITY_CODE3	63
APL_RANGE_DIFF	91
APL_SYMBOL_MOTL	114
APL_SYMBOL_MOTREVL	111
APL_SYMBOL_MOTSPDCL	112
APL_SYMBOL_MOTSPDL	113
APL_SYMBOL_VLVMOTL	110
APL_TIME_OP_DISPLAY	22
APL_TIME_OP_DISPLAY2	23
APL_TIME_OP_DISPLAY3	29
APL_TIME_RANGE	90
APL_TIME_RANGE2	92