SIEMENS

SIMATIC

Process Control System PCS 7 Programming Instructions for Blocks

Manual

Edition 04/2005
A5E00391619-02

Safety Guidelines

This manual contains notices intended to ensure personal safety, as well as to protect the products and connected equipment against damage. These notices are highlighted by the symbols shown below and graded according to severity by the following texts:

Danger
indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

Warning
indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

Caution
indicates that minor personal injury can result if proper precautions are not taken.

Caution
indicates that property damage can result if proper precautions are not taken.

Notice
draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only **qualified personnel** should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:

Warning
This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are registered trademarks of SIEMENS AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

Preface

Purpose of the Programming Instructions

These Programming Instructions describe how to create PCS 7-compliant PLC blocks or faceplates.

The essential differences between a PCS 7-compliant PLC block and a straightforward S7 block are as follows:

- The option of **monitoring** parameter values in a faceplate
- The option of **controlling** parameter values and therefore the way the block executes in a faceplate
- The option of sending **messages** relating to asynchronous events and block states to the OS and to display these messages in a faceplate or a WinCC message list.

Audience

These programming instructions are intended for developers of automation blocks (PLC blocks) and/or faceplates that will be used and fully integrated in the same systems as the PCS 7 process control blocks supplied by Siemens.

Requirements

To use these programming instructions, you therefore require experience in the development and application of PLC blocks and faceplates and should be familiar with the relevant hardware and software. These instructions describe only the measures necessary to achieve conformity between blocks you have created yourself and the PCS 7 blocks. Where necessary, you will find further information in the documentation listed in the References at the end of this manual.

You will find general information on the use of PCS 7 components in the PCS 7 Configuration Manual.

General Outline

These programming instructions provide you with an overview of the individual components of a PCS 7-compliant block. The order in which they are introduced is the same order you would follow to develop function blocks and faceplates.

- You develop the "CONTROL" PLC block, a simple controller block, step by step by first defining the block header, the parameters of the block and its local variables. You then create the source code.
- The next step is to develop a faceplate. You create this with the WinCC Graphics Designer and the elements of the Faceplate Designer.
- The last step is to develop an online help system for the block and then a shippable library MYLIB made up of all the components.

As you work through the instructions, you will see the sections of the sample block required to understand the current topic. Section 1.11 contains a printout of the entire sample PLC block.

In the appendix, you will find a sample source code for MEAS_MON, MOTOR and VALVE blocks contained in the PCS 7 library, as an example of PCS 7-compliant blocks. You can use this source code - or part thereof - as a template for your own blocks.

Note

The use of the sample source code contained in the appendix is the responsibility of the user. There is no guarantee for error-free display.

Further Support

If you have any technical questions, please get in touch with your Siemens representative or agent responsible.

You will find your contact person at:

http://www.siemens.com/automation/partner

You will find a guide to the technical documentation offered for the individual SIMATIC Products and Systems here at:

http://www.siemens.com/simatic-tech-doku-portal

The online catalog and order system is found under:

http://mall.ad.siemens.com/

Training Centers

Siemens offers a number of training courses to familiarize you with the Process Control System PCS 7 automation system. Please contact your regional training center or our central training center in D 90327 Nuremberg, Germany for details:

Telephone: +49 (911) 895-3200.

Internet: http://www.sitrain.com

Technical Support

You can reach the Technical Support for all A&D products

- Via the Web formula for the Support Request
  http://www.siemens.com/automation/support-request
- Phone: + 49 180 5050 222
- Fax: + 49 180 5050 223

Additional information about our Technical Support can be found on the Internet pages http://www.siemens.com/automation/service

Service & Support on the Internet

In addition to our documentation, we offer our Know-how online on the internet at:

http://www.siemens.com/automation/service&support

where you will find the following:

- The newsletter, which constantly provides you with up-to-date information on your products.
- The right documents via our Search function in Service & Support.
- A forum, where users and experts from all over the world exchange their experiences.
- Your local representative for Automation & Drives.
- Information on field service, repairs, spare parts and more under "Services".

Contents

1 Creating AS Blocks

1.1 Requirements and Previous Experience

The blocks described here are intended for use with PCS 7 V6.1 and higher on an S7-4xx CPU. To create the blocks, you require the following software packages:

- STEP 7 standard, V5.2 or higher
- SCL Compiler, V5.1 SP4
- CFC, V6.0

AS blocks for PCS 7 are created with the SCL programming language. This is therefore the only method described in this manual. For further information on SCL, refer to the following:

- The online help in the SIMATIC Manager
  (start help on the optional packages > Programming Blocks with S7-SCL).
- The "S7-SCL for S7-300 and S7-400" manual

These manuals are installed on your hard disk (**Start > Simatic > Documentation**).

1.1.1 Supplied Sample Block

The "CONTROL" block described here is supplied as the SCL source file "S7_CONTx.SCL" with PCS7TOOLS and installed in

...\STEP7\EXAMPLES\zdt25_01\S7_CONTA.SCL (German)
...\STEP7\EXAMPLES\zen25_01\S7_CONTB.SCL (English)
...\STEP7\EXAMPLES\zfr25_01\S7_CONTC.SCL. (English, with French installation)

To include the block in your project, follow the steps outlined below:

1. Select the source folder in your project and then select **Insert > External Source...**. In the "Insert External Source File" dialog box, navigate through the folder structure to the location of the SCL source file, select the file, and then click "Open".

The SCL source file is now in the source folder and must be compiled with the SCL compiler.

2. Before you compile, make sure that the "OP_A_LIM" block (FB 46) is located in the block folder of your project. If this is not the case, copy it to your project from the "PCS 7 Library V61" library.
3. Verify the default settings of the SCL compiler (see Section 1.2.1).
4. Enter the symbolic name "CONTROL" and FB501 as the address and then save the entries.
   (The numbers from 500 upwards are recommended in order to avoid collisions with the numbers of the PCS 7 standard blocks).
   See also the "Entry in the Icon Table" section in chapter 1.2.2.
5. Double-click the "S7_CONTA or S7_CONTB" SCL source file to open it, start the compilation, and close the SCL compiler again after error-free compilation.

The sample block **FB 501** is now located in the block folder of your project.

1.2 Structure of a AS Block

A AS block can only run correctly in the PCS 7 environment if it meets certain formal criteria and has a certain minimum content. The following sections describe what is necessary to meet these criteria.

The block diagram shown below shows the basic structure of the "CONTROL" block (FB 501). The individual elements of the block are explained in greater detail in the sections shown in brackets.

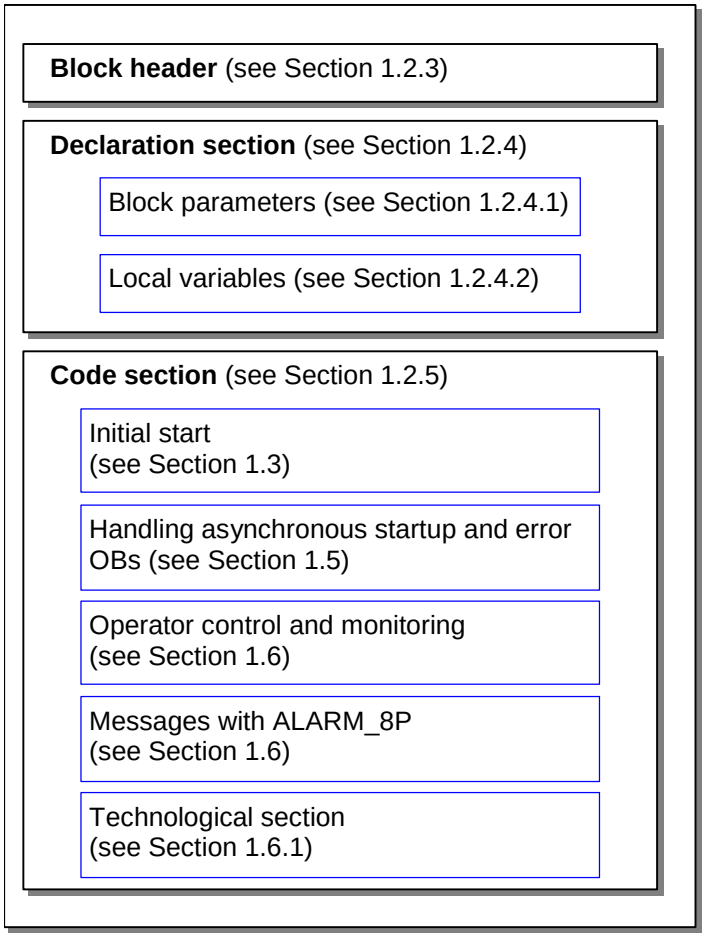


Figure 1-1: Structure of the "CONTROL" (FB 501) sample block

Function Block or Function?

If you want your block to save values, signal messages, or to be available for operator control and monitoring, you must implement it as a function block (FB). An FB has a "memory" in the form of a data block (DB), also known as instance data.

If you do not require this, you can implement your block as a function (FC).

1.2.1 Settings for the SCL Compiler

Settings for "Creating Blocks"

In the SCL compiler, you can set the following options with **Options > Customize > Create Block**:

- **Overwrite blocks**

  Existing blocks in the "Blocks" folder of an S7 program are overwritten if blocks with the same name are created during compilation.

  Blocks with the same name on the AS are also overwritten when you download the blocks.

  If you do not select this option, you must confirm a message before the block is overwritten.

- **Display warnings**

  This option decides whether warnings are also indicated along with errors following compilation.

- **Display errors before warnings**

  This option decides whether errors are listed before warnings in the message.

- **Generate reference data**

  Select this option if you want reference data to be generated automatically when a block is created.

  With the **Options > Reference Data** menu command, you can nevertheless create or update the reference data later.

- **Include system attribute "S7_server"**

  Select this option if you want the system attribute for the "S7_server" parameter to be taken into account when a block is created. You assign this attribute if the parameter is relevant for connection or message configuration. The parameter contains the connection or message numbers.

Note for this sample block:

You must select the "Include system attribute 'S7_server'" check box since this block contains messages. Otherwise, importing or inserting the block in a CFC chart would be stopped with an error message.

Settings for "Compiler"

You can select or clear the following three check boxes in the dialog in the SCL Compiler displayed with **Options > Customize > Compiler**:

- "Monitor array limits"
- "Create debug info"
- "Set OK flag"

The other check boxes should remain selected. For more detailed information on the individual options, refer to the SCL manual.

When you decide whether or not to select these options, remember the following points:

- **Monitor array limits**

  If you use Arrays in the program, during run time a check is made to determine whether the index and declared array lengths are within the permitted range. An error changes the OK flag and the ENO output is reset. This check takes up a considerable amount of run time.

  If you use arrays, you should only leave this option selected until you have adequately tested your block and are sure that the index and array length match.

- **Create debug info**

  This option allows a test to be made with the debugger after the program has been compiled and downloaded to the CPU. This option does, however, increase the memory required for the program and the run times on the AS. You should therefore only select this option during the test phase of the block but not include it in the final version.

- **Set OK flag**

  The OK flag is a system-internal variable. If an error occurs during execution of an option, for example overflow with arithmetic operations, the OK flag is changed by the system and passed to the ENO output. This check takes up a considerable amount of run time. It is therefore advisable to disable the automatic setting of the OK flag and to detect illegal operations/limit violations in the block algorithm yourself. If an error occurs, you can then set the OK flag explicitly if you want to interconnect the ENO output. (This is handled by the system and does not mean any loss in performance since the state of the OK flag is always passed to the output by the system.)

1.2.2 Settings in the SIMATIC Manager

In a PCS 7-compliant AS block, the interface to the user (parameter names, comments etc.) should be in English. You can develop the blocks themselves in any language.

Selecting the Language

If you want to put your blocks together in a library (see also Chapter 4.1), the language must also be set to English so that the folders in your library have PCS 7-compliant names (**Sources**, **Icons** and **Blocks**). To achieve this, you must set "English" for the language and for the mnemonics in the SIMATIC Manager with **Options > Customize > Language**.

Entry in the Icon Table

The name of the block to be entered in the block header (as described below) must be entered in the icon table as the symbolic name.

1. To make this entry, open the icon table in the S7 program by double-clicking "Icons".
2. Enter the symbolic name (here: "CONTROL") in the "Icon" column.
3. Assign an FB number to it in the "Address" column (here: FB 501).
4. Enter a comment in the "Comment" column to describe the block function in closer detail (max. 80 characters).

   This comment represents a supplement to the symbolic block name, since the latter usually does not present a sufficient amount of information about application features or functionality of this block.
   This block comment is identical to icon comment that is displayed in SIMATIC Manager (in the detail view or in the object properties of the block).

   This text is displayed as block comment in the instance block of the block inserted into a CFC chart and, irrespective of the entry in the icon table, it can be adapted specifically to the instance.
   Note: Only a correspondingly limited number of characters is displayed, which depends on the depth of information provided by the CFC blocks. The full comment, however, is displayed temporarily in the short information (“mouse over block header” function).
5. Save and close the icon table.

See also Section 1.10, Naming Conventions and Numeric Range.

1.2.3 Block Header

The block header contains the management information (known as block attributes) of the block. These attributes are used by the PCS 7 tools for various purposes. They are shown in the object properties of the block in the SIMATIC Manager and can also be changed there (see also KNOW_HOW_PROTECT attribute).

Excerpt from the Sample Block:

```
//Copyright (C) Siemens AG 1999. All Rights Reserved. Confidential
(*******************************************************************************
BRIEF DESCRIPTION:
  This block is a sample showing how to develop a PCS 7-compliant
  AS block.

  It implements a simple control algorithm according to the formula:
  Manipulated variable = gain * (setpoint - actual value)

  If the process value exceeds the upper alarm limit, the error output
  QH_ALM is set. A message to the OS is also generated with ALARM_8P.
  The message can be suppressed with the M_SUP_AH variable.

  If the process value falls below the lower alarm limit, the error
  output QL_ALM is set. A message to the OS is also generated with ALARM_8P.
  The message can be suppressed with the M_SUP_AL variable.

  The block supports SIMATIC BATCH and therefore has the required parameters
  BA_EN, BA_NA, BA_ID, OCCUPIED and STEP_NO.

  To indicate a time delay, the block has additional inputs:
  The SUPP_OUT output tracks the SUPP_IN input after a selectable wait time
  SUPPTIME.
*******************************************************************************)
//Author: ABC               Date:               13.08.00   Vers.:1.00
//Modified:                 Date:               18.11.03   Vers.:
//Change:

//*****************************************************************************
// Block header
//*****************************************************************************
FUNCTION_BLOCK "CONTROL"
TITLE =        'CONTROL'
{  // List of system attributes
S7_tasklist:=   'OB80,OB100';  // Block called if a time error occurs and
                               // at warm restart
S7_m_c:=        'true';        // Block can be controlled and monitored
S7_alarm_ui:=   '1'            // Setting for PCS 7 message dialog
                               // ('0'=standard message dialog)
}
AUTHOR:           ABC
NAME:             CONTROL
VERSION:          '0.02'
FAMILY:           XYZ
KNOW_HOW_PROTECT
```

The following two screenshots show the object properties of the compiled sample block with arrows showing the attributes of the block header.

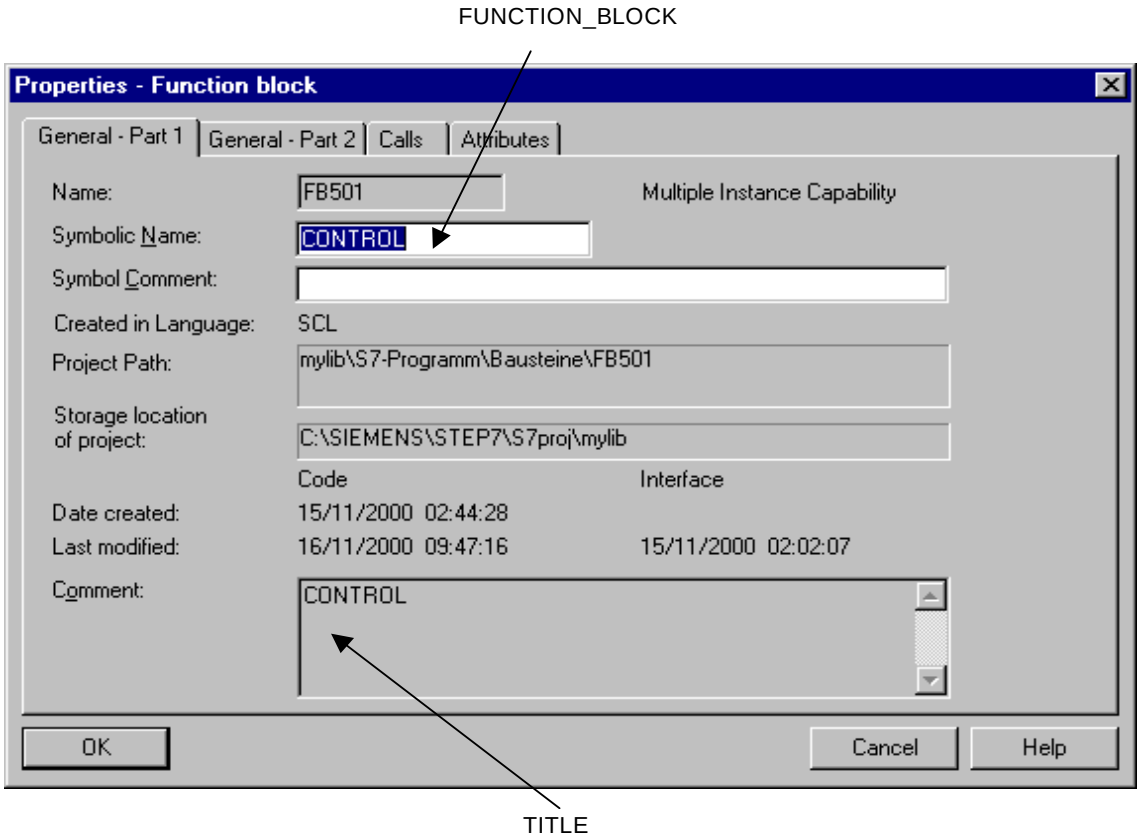


Figure 1-2: Object Properties of the Block (General - Part 1)

- **FUNCTION_BLOCK**

  Here you specify the name of the block with a maximum of 8 characters. This name is displayed in the object properties of the block, in the detailed display in the SIMATIC Manager, and in the CFC catalog. Before you compile the block, a block number must be assigned to this name in the icon table.

- **TITLE**

  This information is not evaluated in PCS 7, however, it is displayed in the SIMATIC Manager in the object properties of the block in the comment field. Comments entered directly below this attribute are also displayed in the object properties of the block in the comment box. All the other comments in the block header can only be seen using the SCL editor.

  It is advisable to enter a short description of the block.

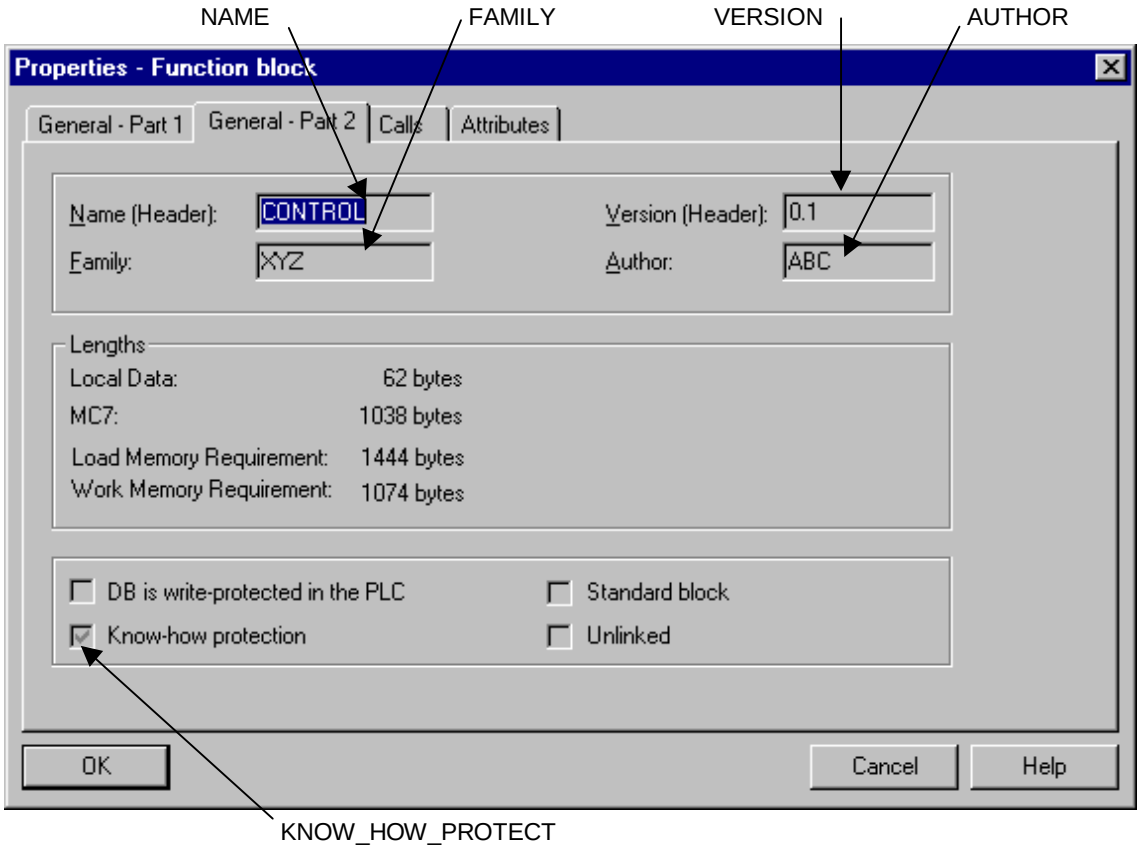


Figure 1-3: Object Properties of the Block (General - Part 2)

- **NAME**

  Here, enter the same name as for FUNCTION_BLOCK. If you intend to use an online help, this name (and FAMILY) forms part of the search string for locating the help text of this block in the help file.

- **VERSION**

  Here, enter a version identifier from 0.00 to 15.15.

- **FAMILY**

  If you want to gather your blocks in a separate library, arrange the blocks in various groups and enter a family name for the block with a maximum of 8 characters.

  If you intend to use an online help, the FAMILY and NAME form part of the search string for locating the help text of the block in the help file (see also Section. 3.2).

- **AUTHOR**

  This attribute normally contains the name or department of the author of the block. In PCS 7-compliant blocks, this is also used for two further purposes:

  - If you want to file all your blocks in one library, enter a common name for all blocks of the library with a maximum of 8 characters.
  - If you use an online help, the relevant help file is located using this name.

- **KNOW_HOW_PROTECT**

  By means of this attribute, you can read- and write- protect the algorithm and the attributes of the block. If this attribute is set, the attributes of the block are only displayed in the SIMATIC Manager in the object properties of the block but they cannot be modified. Outside your project, the block itself can only be opened without the corresponding source file using the STL editor and can no longer be opened with SCL. Even in this case, only the block parameters are displayed. In your own project, the SCL compiler is started.

- **List of System Attributes for Blocks**

  Using the system attributes, you prepare a block for the connection with the OS. The attribute: **S7_m_c**, for example, specifies whether or not the block is relevant for an OS; in other words, whether internal data structures will be created for it on the OS. You can also control the installation of the block in an SFC chart using the system attributes. The attribute S7_tasklist, for example, specifies the OBs in which the block will be installed automatically.

Table 1-1: System Attributes for PCS 7-Compliant Blocks

System Attribute	Meaning	Default
S7_tasklist	Contains a list of the OBs (for example error or startup OBs) in which the block will be installed by CFC.	Not installed more than once
S7_m_c	Specifies whether the block can be controlled or monitored from an OS.	false
S7_alarm_ui	Identifier for message server: S7_alarm_ui :='0' standard message dialog S7_alarm_ui :='1' PCS 7 message dialog	S7_alarm_ui :='0'
S7_tag	If this system attribute has the value 'false', the block is not entered in the tag list of the OS; in other words, it cannot be selected for "Loop in Alarm" on the OS. This is useful for blocks that only send messages but do not have a faceplate. If the system attribute does not exist, the block is entered in the process tag list if it also has the system attribute S7_m_c.	true
S7_driver	ID for the signal preprocessing driver block, which is automatically interconnected with the corresponding block by means of the CFC function „Generate module drivers“ in SIMATIC Manager. Values: 'chn', 'f'	- -
S7_hardware	ID for a module-specific driver block, which is automatically inserted into the CFC chart, configured and interconnected by means of the CFC function „Generate module drivers“ in SIMATIC Manager.. Values: 'subnet', 'rack', 'sm', 'im', 'fm'	- -
S7_read_back	Defines whether the block instance is to be allocated to the "Chart > Readback" function in the CFC. The instance block parameters cannot be read back when the value of this system attribute is 'false'.	true

The system attributes are displayed in the object properties of the block in the "Attributes" tab in the SIMATIC Manager where they can also be modified if the block is not write-protected (KNOW_HOW_PROTECT attribute in the block header).

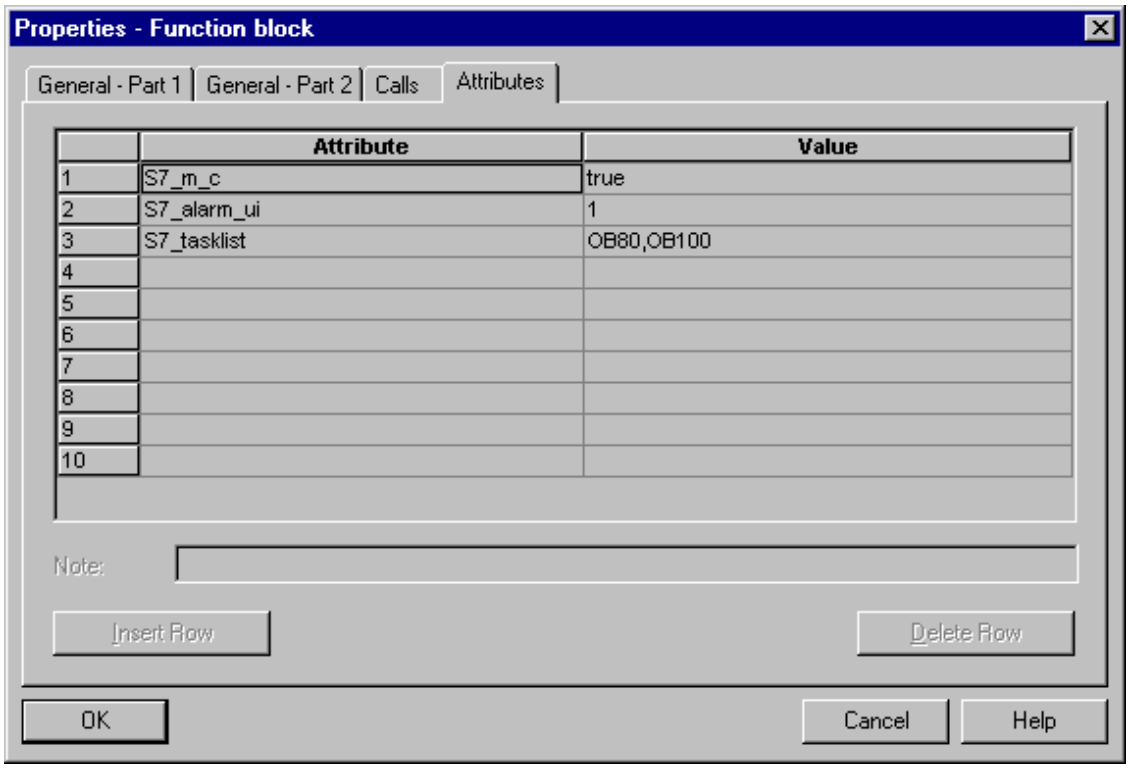


Figure 1-4: System Attributes of the Block

Note:

You can display a complete list of system attributes in the context-sensitive help and/or in the "Attributes for Blocks and Parameters" help topic.

1.2.4 Declaration Section

1.2.4.1 Block Parameters

The block parameters define the interface of the block to other blocks and to the operator control and monitoring tools (CFC, WinCC ...).

Parameter Types

The following parameter types exist:

- **Input parameters**

  Input parameters must be specified for PCS 7-compliant blocks in the following situations:

  - You want to fetch parameter values from another block or
  - You want to control parameters at the OS
  - You want to be able to influence the display of a faceplate on the OS using parameters (for example limits for the displayed ranges)
  - You want to control parameters for test purposes when working in CFC
  - You want to set parameters to generate messages (message event ID of the ALARM_8P block)
  - If you require a smooth changeover between input values from the program (CPU) and operator input values (OS), the input parameters can be both read and written back (see inout parameters) by the block algorithm. In contrast to the inout parameters, the input parameters are not written back to the interconnected output parameters.
  - **Output parameters**

  Output parameters must be specified for PCS 7-compliant blocks in the following situations:

  - You want to pass parameter values to another block
  - You want to monitor parameters at the OS
  - You want to monitor parameters for test purposes when working in CFC

- **In/out parameters**

  In/out parameters can be both read and written back by the block algorithm. In/out parameters must be specified for PCS 7-compliant blocks when you require a bumpless switchover between input values from the program (AS) and values entered by the operator (OS). To implement this functionality, you require three parameters:

  - An input parameter for switchover
  - An input parameter for the connected value
  - An in/out parameter for the value input by the operator. This parameter must be an in/out parameter since the connected value must always be written back to the value input by the operator. This ensures that the switchover from the connected to the operator value is bumpless.

Comments for Parameters

If you want to add comments to the block parameters, these must be preceded by "//" after the parameter definition.

Comments are displayed in the CFC chart in the object properties of the I/O and in the object properties of the block in the "Inputs/Outputs" tab. They can also be modified here regardless of the block protection (KNOW_HOW_PROTECT attribute in the block header).

System Attributes for Parameters

Just like the block itself, block parameters can also be specified in greater detail using system attributes.

This allows you to define the following:

- How the parameter is displayed on the OS.
  Example: **S7_unit** defines the unit of the parameter (for example liters). You can also display the text specified for this attribute in your faceplate.
- Whether and how the parameter is handled in CFC.
  Example: **S7_visible** defines whether or not the parameter is displayed in the CFC chart.

Table 1-2: System Attributes for Parameters for PCS 7-Compliant Blocks

System Attribute	Affects	Meaning	Default
S7_sample-time	Time response	If a parameter has this system attribute it is automatically assigned the cycle time of the calling cyclic OB. When you compile the CFC chart, the "Update sampling time" check box must be selected (see also Section 1.4).	false
S7_dynamic	CFC	If a parameter has this system attribute, it is automatically registered for testing in the test mode of CFC.	false
S7_edit	CFC	This decides whether or not the parameter can be edited in the SIMATIC Manager in the "Edit Parameters/Signals" table (without opening the CFC chart).	false
S7_link	CFC	This decides whether or not the parameter can be interconnected in the CFC chart.	true
S7_param	CFC	This decides whether or not the parameter value can be set in the CFC chart.	true
S7_visible	CFC	If this system attribute is set to 'false' for a parameter, it is not displayed in the CFC chart.	true
S7_qc	CFC, OCM	This attribute identifies parameters, in addition to process values, that also have quality codes. The quality code has to be declared as the next direct parameter after the process value in the interface. The data type of the process value can be any, but the quality code has to be of the data type "BYTE".	false
S7_contact	SFC	The parameter belongs to an exclusion group	false
S7_m_c	OCM	This decides whether or not the parameter can be controlled or monitored from an OS.	false
S7_shortcut	OCM	This contains a maximum 16 character long identifier for the parameter. This name (for example "Setpoint"), can also be displayed in a faceplate on the OS.	- -

System Attribute	Affects	Meaning	Default
S7_string_0	OCM	This system attribute is only relevant for input parameters (or in/out parameters) of the data type BOOL. It contains a text with a maximum of 16 characters that can be displayed in a faceplate as an operator text (for example "Open Valve"). When the operator selects this function, the parameter is given the value 0.	- -
S7_string_1	OCM	This system attribute is only relevant for input parameters (or in/out parameters) of the data type BOOL. It contains a text with a maximum of 16 characters that can be displayed in a faceplate as an operator text (for example "Close Valve"). When the operator selects this function, the parameter is given the value 1.	- -
S7_unit	OCM	This contains the unit of the parameter and can have a maximum of 16 characters. The unit (for example "mbar") can be displayed in CFC at the block I/O.	- -
S7_server	Server	The interface parameter is assigned to a server. Message server: S7_server:='alarm_archiv'	No server call
S7_a_type	No server call	The interface parameter is the message number input of message type x or archive number input	- -

Using and Modifying System Attributes

When using the system attributes "S7_string_0" and "S7_string_1", remember the following points:

The specified value can be displayed in the faceplate as an operator text. If the operator executes the function, the value 0 or 1 is transferred to the AS. In CFC, the current value of the parameter is displayed. You can also adapt this value output using the system attribute.

To do this, you divide the system attribute into two parts separated by an equality character, for example, S7_string_1 := 'Suppress HH =YES'. CFC recognizes the equality character and replaces the value output of the parameter by the section following the equality character; in other words, in this case instead of the value 1 the word "YES" is output.

In CFC, a maximum of 8 characters are displayed even if you specify more. In the faceplate, the entire text is always displayed, in this case, "Suppress HH =YES".

In CFC, the system attributes are displayed in the object properties of the I/O and can also be modified there. The following screenshots show the object properties of parameters of the data type BOOL and parameters that are not of the data type BOOL (Figure 1-6). The screenshots include arrows indicating the relevant system attributes.

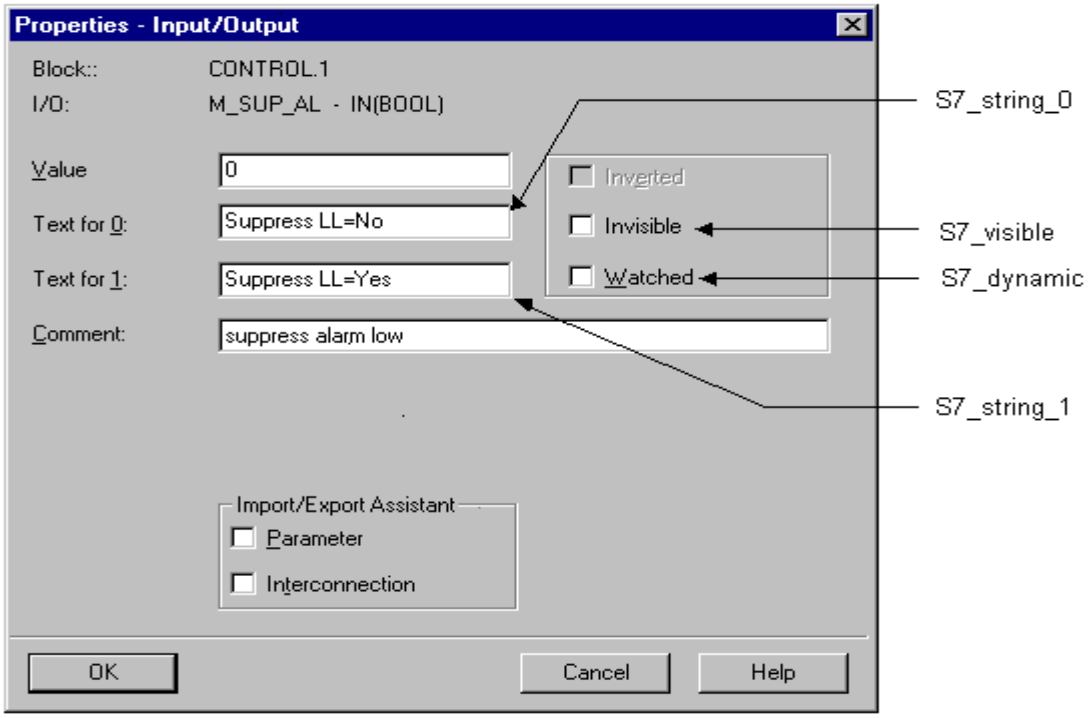


Figure 1-5: Object Properties of Parameters of the BOOL Data Type

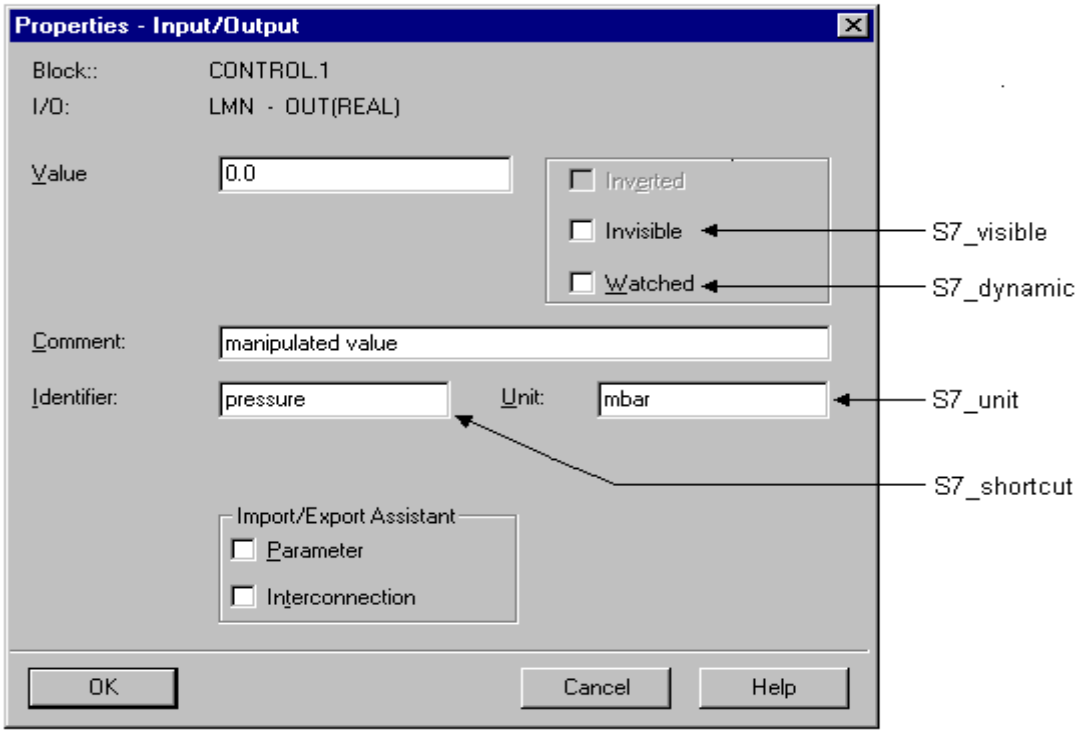


Figure 1-6: Object Properties of Parameters not of the BOOL Data Type

The following excerpt from the sample block shows the coding of the block parameters:

```
//****************************************************************************//
Declaration Section: Block Parameters
//****************************************************************************
VAR_INPUT
SAMPLE_T {S7_sampletime:= 'true';// Param.of block sampling time (cycle of task)
       S7_visible:='false';        // Parameter not visible
       S7_link:= 'false'           // Parameter cannot be linked
      }          :REAL := 1;       // sample time [s] (default 1 sec)

H_ALM {S7_m_c := 'true';
    S7_visible:='false';
    S7_link := 'false'}  :REAL :=100;  // upper alarm limit (default 100)

L_ALM {S7_m_c := 'true';           // Parameter has op cont and mon capability
     S7_visible:='false';          // Parameter not visible
     S7_link := 'false'            // and cannot be linked
     }           :REAL := 0;       // lower alarm limit (default 0)

M_SUP_AL {S7_visible:='false';
S7_link:='false';
S7_m_c:='true';
    S7_string_0:= 'Suppress LL=No';    // Operator text for value (M_SUP_AL)= 0
    S7_string_1:= 'Suppress LL=Yes'    // Operator text for value (M_SUP_AL)= 1
      }           :BOOL;               // suppress alarm low

M_SUP_AH {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_string_0:= 'Suppress HH=No';
    S7_string_1:= 'Suppress HH=Yes'
      }           :BOOL;              // suppress alarm high

SP_OP_ON {S7_visible:='false';
    S7_dynamic:='true' // CFC in test: display of actual value in AS)
    }           :BOOL := 1; // Enable 1=operator for setpoint input

SPBUMPON {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_string_0:='SP bumpless=Off';
    S7_string_1:='SP bumpless=On'
   }
          :BOOL := 1;             // Enable 1=bumpless for setpoint on
SP_EXTON {S7_visible:='false';
    S7_dynamic:='true'             // CFC in test/commissioning: display of
   }                               // actual value in AS)
                  :BOOL := 1;      // 1: Select SP_EXT
SP_EXT   {S7_dynamic:='true'}
                  :REAL := 0;      // External setpoint

SP_HLM   {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_shortcut:='SP high limit';  // Text (max 16 chars) for display on OS
    S7_unit:=''}                   // Unit (max 16 chars)
               :REAL := 100;       // Setpoint high limit

SP_LLM    {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_shortcut:='SP low limit';
    S7_unit:=''}      :REAL := 0;  // Setpoint low limit
PV_IN {S7_dynamic:='true';
  S7_m_c:='true';
  S7_unit:='%'} :   REAL := 0;     // Process value (to AUX_PR04 of message)

GAIN   {S7_link:='false';
```

```
    S7_edit:='para';     // Parameter setting in Import/Export Assistant (IEA)
    S7_m_c:='true';
    S7_shortcut:='Gain';
    S7_unit:=''}        :REAL := 1; // Proportional gain

EV_ID   {S7_visible:='false';
    S7_link:='false';
    S7_param :='false';             // Parameter cannot be set in CFC
    S7_server:='alarm_archiv';      // Message no. assigned by server
    S7_a_type:='alarm_8p'           // Block signals with ALARM_8P
       }           :DWORD := 0;     // Message ID

                                    // Parameters for SIMATIC BATCH
STEP_NO {S7_visible := 'false';
    S7_m_c   := 'true'}  :DWORD;     // Batch step number
BA_ID {S7_visible  := 'false';
    S7_m_c    := 'true'}  :DWORD;   // Batch ID
BA_EN {S7_visible := 'false';       // Parameter not visible in CFC chart
    S7_m_c := 'true'           // Parameter has op cont and mon capability
    }             :BOOL := 0;  // Batch enable
BA_NA {S7_visible := 'false';
    S7_m_c   := 'true'} :STRING[32] := '';  // Batch name

OCCUPIED {S7_visible := 'false';
    S7_m_c   := 'true'} :BOOL := 0;         // Occupied by Batch

RUNUPCYC   {S7_visible:='false';
    S7_link:='false'} :INT := 3;     // Number of run up cycles
SUPPTIME       :REAL := 0;           // Sample delay
SUPP_IN        :REAL := 0;           // Input value for sample delay
END_VAR

VAR_OUTPUT
  LMN {S7_shortcut:='pressure';      // Name of the parameter on OS
    S7_unit := '%';                 // Unit of the parameter
    S7_m_c := 'true'                // Can be monitored
    } :REAL;                        // Manipulated value

  QH_ALM  :BOOL := false;           // 1 = HH alarm active

  QL_ALM  :BOOL := false;           // 1 = LL alarm active

  QSP_HLM   {S7_visible:='false';
    S7_dynamic:='true'} :   BOOL := 0;     // 1=Setpoint output high limit active

  QSP_LLM   {S7_visible:='false';
    S7_dynamic:='true'} :   BOOL := 0;     // 1=Setpoint output low limit active

  Q_SP_OP   {S7_visible:='false';
    S7_dynamic:='true';
    S7_m_c:='true'} :   BOOL := 0;    // Status: 1=Operator may enter setpoint

  QOP_ERR   {S7_visible:='false';
    S7_dynamic:='true'} :   BOOL := 0;   // 1=Operator error

  QMSG_ERR   {S7_visible:='false';
    S7_dynamic:='true'} :   BOOL := 0;   // ALARM_8P: Error output

  MSG_STAT   {S7_visible:='false';
    S7_dynamic:='true'} :   WORD := 0;   // Message: STATUS output

  MSG_ACK   {S7_visible:='false';
    S7_dynamic:='true'} :   WORD := 0;   // Message: ACK_STATE output

  SUPP_OUT           :REAL := 0;         // Output value for sample delay
  SP   {S7_dynamic:='true';
    S7_m_c:='true'} :    REAL := 0;      // Active setpoint

END_VAR
```

```
VAR_IN_OUT
SP_OP {S7_visible:='false';
  S7_link:='false';
  S7_m_c:='true';
  S7_shortcut:='Setpoint';
  S7_unit:=''} :   REAL := 0;     // Operator input setpoint

// Freely assignable auxiliary values of ALARM_8P
AUX_PR05 {S7_visible := 'false'} : ANY;  // Auxiliary value 5
AUX_PR06 {S7_visible := 'false'} : ANY;  // Auxiliary value 6
AUX_PR07 {S7_visible := 'false'} : ANY;  // Auxiliary value 7
AUX_PR08 {S7_visible := 'false'} : ANY;  // Auxiliary value 8
AUX_PR09 {S7_visible := 'false'} : ANY;  // Auxiliary value 9
AUX_PR10 {S7_visible := 'false'} : ANY;  // Auxiliary value 10
END_VAR
```

1.2.4.2 Local Variables

Additional variables that are not output in any way as block parameters must be defined as local variables.

There are two types of local variables:

- Static variables
- Temporary variables

Static Variables

Static variables, in contrast to temporary variables, retain their value over multiple block calls until you change the value in the block algorithm.

In PCS 7-compliant blocks, these variables are particularly important if you want to call existing blocks, either your own or standard blocks, within your block. In this case, you must implement a **multiple instance** block. This is done by defining an instance of the called block within the static variables.

Before the calling block can be compiled free of errors, the called blocks must exist in the block folder of the S7 program.

If you want to visualize and interconnect parameters of the called block, you must copy these from your block algorithm or into parameters of your block. The parameters of the called block itself are not visible to the outside world.

Multiple Instances

You will find examples of multiple instance applications in the section on the CFC block types and in the relevant SCL code in the sample project.

Note:

Called SFBs and SFCs, such as SFC 6 (RD_SINFO) or SFB 0 (CTU) are found automatically in the standard library and entered in your S7 program when you compile the calling block.

Called FBs are copied to the block folder when you insert the calling block in a CFC chart if they are located in the same library as the calling block. Otherwise, you must copy them yourself.

Temporary Variables

Temporary variables are valid only during **one** block call; in other words, they must be recalculated at each block call.

Here, there are no special rules for PCS 7-compliant blocks.

Excerpt from the Sample Block:

```
//*****************************************************************************
// Declaration Section: Temporary Variables
//*****************************************************************************

VAR_TEMP
// Start info: Structure with info for the OB that has just called the block
TOP_SI:    STRUCT
  EV_CLASS :BYTE;
  EV_NUM  :BYTE;
  PRIORITY :BYTE;
  NUM    :BYTE;
  TYP2_3  :BYTE;
  TYP1   :BYTE;
  ZI1    :WORD;
  ZI2_3   :DWORD;
END_STRUCT;

// Start info: Structure with info for the last called startup OB
START_UP_SI:  STRUCT
  EV_CLASS :BYTE;
  EV_NUM  :BYTE;
  PRIORITY :BYTE;
  NUM    :BYTE;
  TYP2_3  :BYTE;
  TYP1   :BYTE;
  ZI1    :WORD;
  ZI2_3   :DWORD;
END_STRUCT;
  S7DT   :DATE_AND_TIME;    // Local time variable
  DUMMY  :INT;              // Auxiliary variable
END_VAR
```

1.2.5 Code Section

The code section contains the actual algorithm of the block. With PCS 7-compliant blocks, this means that you must implement not only the technological functions of the blocks but also the properties, for example, to signal asynchronous events and block states to the OS and to display them on the OS in a faceplate or WinCC message list.

1.3 Initial Start

When your block is called for the first time, various parameters normally need to be initialized. Depending on the technological function of your block, there may also be other activities that your block needs to execute once. If this is the case in your block, you must implement an initial start section.

You do this by defining a variable of the BOOL data type (for example sbRESTART). You can implement the variables as static variables.

Since there is no guarantee that your block will be executed for the first time only during a restart (for example after the block has been downloaded again with the CPU in the RUN mode), it is normally necessary to integrate the initial start section in the cyclic part of your block. This allows you to extend the execution of the initial start, when necessary, over several call cycles of the block.

```
//*****************************************************************************
// Dependence on Calling OB
//*****************************************************************************

// Read out start info with SFC6 (RD_SINFO)
  DUMMY := RD_SINFO (TOP_SI := TOP_SI, START_UP_SI := START_UP_SI);
  IF sbRESTART THEN
  // Initial start
  TOP_SI.NUM := 100;   // Execute initial start as warm restart
  sbRESTART := FALSE;  // Reset initial start
  END_IF;
// In which OB was the block called ?
CASE WORD_TO_INT(BYTE_TO_WORD(TOP_SI.NUM)) OF

//*****************************************************************************
// Handling error OBs
//*****************************************************************************

  // OB80: time error
  80:
  QH_ALM  := 0;  // Reset error outputs
  QL_ALM  := 0;

//*****************************************************************************
// Startup
//*****************************************************************************

  // OB100: Warm restart
  100:
  QH_ALM  := 0;  // Reset error outputs
  QL_ALM  := 0;
  siRUNUPCNT := RUNUPCYC; // Save RUNUPCYC value
ELSE
....
```

1.4 Time Dependencies

If your block is executed in a timed interrupt level in the constant scan time mode and if it needs to evaluate the length of the interval to execute time-dependent activities (for example controller blocks), define an input parameter (for example SAMPLE_T) of the REAL data type at which the length of the time interval can be specified.

Depending on the cyclic interrupt OB in which your block is called, these parameters must be given new values. This makes sure that your block algorithm always operates with the correct time.

If you give these parameters the system attribute **S7_sampletime**, and set this to 'true', CFC automatically sets the value to match the calling OB. A scan rate is also taken into account. You should also give the parameter the system attributes **S7_visible** and **S7_link** and set them to 'false'. The parameter is then invisible and cannot be interconnected in the CFC chart preventing its value being changed accidentally by the user.

Automatic assignment of a value to the parameter by the CFC chart functions, however, only when the "Update sampling time" check box is activated when the program is compiled.

The following section of the sample block illustrates the implementation of a time-dependent action. Using the SUPPTIME parameter, a wait time can be set in the block. Changes at the SUPP_IN input are passed on to the SUPP_OUT output once this wait time has elapsed.

```
//*****************************************************************************
// Declaration Section: Block Parameters
//*****************************************************************************
VAR_INPUT
SAMPLE_T {S7_sampletime:= 'true'     // Block sampling time parameter
                                     //(=cycle of the task)
       S7_visible:='false';          // Parameter not visible
       S7_link:= 'false'             // Parameter cannot be linked
      }        :REAL := 1;           // sample time [s] (default 1 second)
....
END_VAR

//*****************************************************************************
// Declaration Section: Static Variables
//*****************************************************************************
VAR
....
  sSUPP_IN    :REAL := 0;            // Old value of sample delay input
  ACT_TIME    :REAL := 0;            // Time counter
....
END_VAR
VAR_OUTPUT
....
SUPP_OUT           :REAL := 0;       // Output value for sample delay
....
END_VAR

//*****************************************************************************
// Technological Section
//*****************************************************************************

  IF (SUPP_IN <> sSUPP_IN) THEN
    ACT_TIME := SUPPTIME;            // Initialize time counter
    sSUPP_IN := SUPP_IN;
  END_IF;
  IF (ACT_TIME > 0) THEN             // If wait time not yet elapsed
    ACT_TIME := ACT_TIME-SAMPLE_T;   // Count down wait time
  ELSE
    SUPP_OUT := SUPP_IN;             // Connect input to output
  END_IF;
....
```

1.5 Handling Asynchronous Startup and Error OBs

If an asynchronous event occurs, such as warm restart or removing/inserting a module, rack failure and similar events, the AS calls an asynchronous OB. If you want your block to react to such an event, you must install your block in the relevant OB and check whether such an event has occurred in the block algorithm.

Installation in Asynchronous OBs

To install your block in specific OBs, use the "S7_tasklist" system attribute. As the value for this attribute, you then enter all the OBs you require (for example, S7_tasklist := 'OB80, OB100'). When you insert the block in a CFC chart, CFC automatically installs the block in the current cyclic interrupt OB and in all the OBs specified by S7_tasklist.

Checking the Calling OB

To find out the OB in which your block is currently being executed, you must call SFC6 (RD_SINFO) in the block algorithm. This reads the startup information of your block and provides information about the currently active OB (parameter TOP_SI) and the last startup OB to be called (parameter START_UP_SI).

The two parameters have identical structures that must both be defined in your temporary variables. The elements of the structure have the following significance:

Table 1-3: TOP_SI and START_UP_SI Parameters

Structure Element	Data Type	Meaning
EV_CLASS	BYTE	Bits 0 to 3: Event ID Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Priority class number
NUM	BYTE	Number of the calling OB
TYP2_3	BYTE	Data ID for ZI2_3
TYP1	BYTE	Data ID for ZI1
ZI1	WORD	Additional info 1
ZI2_3	DWORD	Additional info 2_3

In terms of their content, the structure elements correspond to the temporary variables of the calling OB. Depending on the OB, however, they can have different names and data types. This means that you must assign the individual structure elements to each other and evaluate them according to the relevant OB description (refer to the "STEP 7 - System and Standard Functions manual). The following table and excerpt from the sample block illustrate this based on the example of OB 80 (time error).

Table 1-4: Assignment of the Elements of the TOP_SI Startup Information to the Temporary Variables of OB 80

TOP_SI / STARTUP_SI		**OB 80**	
Structure Element	**Data Type**	**Temporary Variable**	**Data Type**
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

Note:

- In its temporary variables, each OB contains the date and time of the call. These are, however, not included in the startup information read with SFC6.
- PCS 7-compliant blocks are not installed in the hot restart OB (OB 101) or cold start (OB 102).

The following excerpt from the sample block shows the way in which the OB is handled:

```
//*****************************************************************************
// Code Section
//*****************************************************************************

 CASE WORD_TO_INT(BYTE_TO_WORD(TOP_SI.NUM)) OF

//*****************************************************************************
// Handling error OBs
//*****************************************************************************

  // OB80: time error
  80:
  QH_ALM  := 0;  // Reset error outputs
  QL_ALM  := 0;

//*****************************************************************************
// Startup
//*****************************************************************************

  // OB100: Warm restart
  100:
  QH_ALM  := 0;             // Reset error outputs
  QL_ALM  := 0;
  siRUNUPCNT := RUNUPCYC;   // Save RUNUPCYC value
ELSE
```

1.6 Operator Control and Monitoring and Messages

A block whose parameters can be **controlled** and **monitored** by the operator at the OS must be prepared for this connection to the OS. This involves both the required parameters and the block itself.

Operator Control

If you want to control a parameter value solely from the OS, you require an in/out or input parameter for the value to be controlled (with the system attribute **S7_m_c**).

If, on the other hand, you want the option of either fetching a parameter value from another block or setting the value at the OS and want the switchover from the connected to operator-controlled value to be bumpless, you require a total of three parameters as follows:

- An input parameter for switching over between operator input and interconnected value
- An input parameter for the connected value
- An in/out parameter for the operator-controlled value (with the system attribute **S7_m_c**). This parameter must be an in/out parameter since the interconnected value for bumpless switchover from the block algorithm to the operator-controlled value must be written back as long as the interconnected value is selected.

All the operator input functions should be handled using the operator control blocks of the "PCS 7 V61" library and their corresponding operator input method on the OS. All the required interlocks and the (optionally bumpless) switchover between the operator control value and the interconnected value then exist (for example, for manual/automatic switchover). You can install the operator control blocks in your block using the multiple instance technique.

In PCS 7, for example, the **OP_A_LIM** (**o**peration **a**nalog **lim**ited) block can be used.

The OP_A_LIM block provides you with a limiting reaction to operator input. As an alternative, you can use the OP_A_RJC block that the rejects the operator input if a limit value would otherwise be violated. If you do not require a limit value check, use the OP_A block.

Note:

The execution of a AS block and faceplate is asynchronous; on other words, when an operator enters a value in faceplate, the value is written to the instance DB of the AS block and evaluated later by the AS block. Since the relevant limits may already have changed at this point in time, the operator input value should be checked for errors both on the AS and on the OS.

For binary operator input, you can use the OP_D (FB 48), OP_D3 (FB 49) and OP_TRIG (FB 50) blocks of the PCS 7 Library V61 (for further information, refer to the online help).

The following excerpt shows the definition of an operator input:

```
//*****************************************************************************
// Operator Input of Setpoint SP_OP (Real Value) or Connected Setpoint SP_EXT
//*****************************************************************************

// Multiple instance call OP_A_LIM (for meaning of parameters,
// see online help OP_A_LIM)

     OP_A_LIM_1(U := SP_OP, U_HL:= SP_HLM, U_LL:= SP_LLM, OP_EN:= SP_OP_ON,
BTRACK:= SPBUMPON, LINK_ON:= SP_EXTON, LINK_U:= SP_EXT);

     OK := OK AND ENO; //Enter enable out of OP_A_LIM in OK flag of the block
     Q_SP_OP := OP_A_LIM_1.QOP_EN;            // 1: Enable operator input of SP
     QOP_ERR := OP_A_LIM_1.QOP_ERR;           // 1: Operator error
     QSP_HLM := OP_A_LIM_1.QVHL;              // 1: High limit
     QSP_LLM := OP_A_LIM_1.QVLL;              // 1: High limit
     SP   := OP_A_LIM_1.V;                    // Effective setpoint
```

Messages

If you want your block to send messages and/or events to the OS, you can define a multiple instance of an alarm block in the static variables. The properties of the CPU (selectable "Confirmation-triggered messages “) and of the installed alarm block determine the message and acknowledgment response as well the passing of auxiliary values.

The "Standard Library" contains ready-made alarm blocks as SFBs.
These, for example, include the following:

ALARM	SFB33	Monitoring a signal with 1 to 10 auxiliary values **with** confirmation prompt
ALARM_8	SFB34	Monitoring of up to 8 signals
ALARM_8P	SFB35	Monitoring of up to 8 signals with 1 to 10 auxiliary values
NOTIFY	SFB36	Monitoring a signal with 1 to 10 auxiliary values **without** confirmation prompt
NOTIFY_8P	SFB	Monitoring up to 8 signals with 1 to 10 auxiliary values **without** confirmation prompt

Entries in the Block Header

To allow the block to be controlled and/or monitored at the OS, first set the system attribute "S7_m_c" to 'true' in the list of system attributes in the block header.

To call the PCS 7 message dialog in SIMATIC Manager, enter the attribute S7_alarm_ui := '1' in the block header (if you enter the value '0', the STEP 7-compliant dialog is called).

Entries in the Declaration Section

To allow the parameters of your block to be controlled and monitored at the OS, set the system attribute "S7_m_c" to 'true' for each parameter of your block that you want to control and monitor.

If you want your block to send messages and/or events to the OS, define an input with the DWORD data type (here: EV_ID). In the instance DB, this input adopts the message number automatically assigned by the system (message server).

The message number is unique in the entire S7 project so that there are no collisions in projects involving several CPUs and operator stations. The numbers for individual messages required by WinCC are derived from this message number during data transfer (compile OS).

For this input, specify the system attribute "S7_server" with the value 'alarm_archiv' and the system attribute "S7_a_type" with the value 'alarm_8p' (depending on the message block you have actually installed).
The input should not be visible in the CFC chart and it should not be possible to interconnect or assign parameters to the input to avoid accidentally changing data assigned by the system.

The following excerpt from the sample block illustrates the use of the system attributes in the **block header** and for the **input** EV_ID that will receive the message.

```
//*****************************************************************************
// Block header
//*****************************************************************************

FUNCTION_BLOCK      "CONTROL"
TITLE=              'CONTROL'

{  // List of system attributes
S7_tasklist:=       'OB80,OB100';  // Block is called if there is a time error
                                   // and at a warm restart
S7_m_c:=            'true';        // Block can be controlled and monitored
S7_alarm_ui:=       '1'            // Setting for PCS 7 message dialog
('0'=standard dialog)
}
AUTHOR:             ABC
NAME:               CONTROL
VERSION:            '0.02'
FAMILY:             XYZ
KNOW_HOW_PROTECT

//*****************************************************************************
// Declaration Section: Block Parameters
//*****************************************************************************

VAR_INPUT
                                     // EVENT ID parameter for message no.
  EV_ID   {S7_visible:='false';      // Parameter not visible in CFC
    S7_link:='false';                // Parameter cannot be linked in CFC
    S7_param :='false';              // Parameter cannot be set in CFC
    S7_server:='alarm_archiv';       // Message no. assigned by server
    S7_a_type:='alarm_8p'            // Block signals with ALARM_8P
       }            :DWORD := 0;     // Message ID
...
END_VAR
```

The inputs of the ALARM block not required in the block can be applied to the block interface to allow the later user further options for messages. You can define additional inputs and create logical links in your block algorithm to enable or lock these messages (see 1.6.2). Otherwise, these messages are handled without taking further provisions and can only be disabled by the message system of the OS. This also applies to unused a auxiliary values. You can then use these in the messages as described in Section 1.7.

The following example shows the definition of ALARM_8P:

```
//*****************************************************************************
// Declaration Section: Block Parameters
//*****************************************************************************
....
                            // Freely assignable auxiliary values of ALARM_8P
AUX_PR05 {S7_visible := 'false'} : ANY;  // Auxiliary value 5
AUX_PR06 {S7_visible := 'false'} : ANY;  // Auxiliary value 6
AUX_PR07 {S7_visible := 'false'} : ANY;  // Auxiliary value 7
AUX_PR08 {S7_visible := 'false'} : ANY;  // Auxiliary value 8
AUX_PR09 {S7_visible := 'false'} : ANY;  // Auxiliary value 9
AUX_PR10 {S7_visible := 'false'} : ANY;  // Auxiliary value 10
....

//*****************************************************************************
// Declaration Section: Static Variables
//*****************************************************************************
....
//*****************************************************************************
// Declaration Section Multiple Instances
//*****************************************************************************
  OP_A_LIM_1:  OP_A_LIM;   // Operator control block 1
  ALARM_8P_1:  ALARM_8P;   // Generation of max. 8 messages with
  max. 10 auxiliary values
...
//*****************************************************************************
// Messages with ALARM_8P
//*****************************************************************************

// STRING variables must not be linked to ALARM8_P as auxiliary values
// and are transferred as array of bytes
FOR DUMMY := 1 TO 16
DO
  sbyBA_NA[DUMMY] := 0;  //Delete array as default
END_FOR;
DUMMY := BLKMOV (SRCBLK:= BA_NA,DSTBLK:=sbyBA_NA);
swSTEP_NO  := STEP_NO;        // Batch step number (due to I/O aux val ALARM_8P)
sdBA_ID   := BA_ID;           // Batch ID     (due to I/O aux val ALARM_8P)
srPV_IN   := PV_IN;           // Auxiliary value must not be a INPUT
ALARM_8P_1(EN_R := TRUE,      // Update output ACK_STATE
      ID := 16#EEEE,          // Data channel for messages (always 16#EEEE)
      EV_ID:= EV_ID,          // Message number > 0
      SIG_1:= NOT M_SUP_AH AND QH_ALM,  // Signal 1 to be monitored
                                        // 0 -> high alarm message
      SIG_2:= NOT M_SUP_AL AND QL_ALM,  // Signal 2 to be monitored
                                        // 1 -> low alarm message
      SIG_3:= 0,           // Signal 3 to be monitored -> no message
      SIG_4:= 0,           // Signal 4 to be monitored
      SIG_5:= 0,           // Signal 5 to be monitored
      SIG_6:= 0,           // Signal 6 to be monitored
      SIG_7:= 0,           // Signal 7 to be monitored
      SIG_8:= 0,           // Signal 8 to be monitored
      SD_1 := sbyBA_NA,      // Auxiliary value 1
      SD_2 := swSTEP_NO,     // Auxiliary value 2
      SD_3 := sdBA_ID,       // Auxiliary value 3
      SD_4 := srPV_IN,       // Auxiliary value 4
      SD_5 := AUX_PR05,      // Auxiliary value 5
      SD_6 := AUX_PR06,      // Auxiliary value 6
      SD_7 := AUX_PR07,      // Auxiliary value 7
      SD_8 := AUX_PR08,      // Auxiliary value 8
      SD_9 := AUX_PR09,      // Auxiliary value 9
      SD_10:= AUX_PR10);     // Auxiliary value 10

    QMSG_ERR := ALARM_8P_1.ERROR;     // ERROR status parameter
    MSG_STAT := ALARM_8P_1.STATUS;    // STATUS status parameter
    MSG_ACK := ALARM_8P_1.ACK_STATE;  // Current OS confirmation status
....
```

1.6.1 Message Suppression during Startup

If you want to reduce the load on the AS during startup caused by the simultaneous generation of several messages (by various blocks), define an input parameter RUNUPCYC with the INT data type. With this parameter, you can specify the number of startup cycles during which no message should be generated. In the block algorithm, you then count the number of calls and enable messages only after the set number of cycles has elapsed. The following excerpt from the sample block illustrates how this is done.

```
//*******************************************************************************
// Declaration Section: Block Parameters
//*******************************************************************************
VAR_INPUT
...
H_ALM {S7_m_c := 'true';         // Parameter has op cont and mon capability
   S7_visible:='false';          // Parameter not visible
   S7_link := 'false'            // and cannot be linked
   }  :REAL :=100;        // Upper alarm limit (default 100)

L_ALM {S7_m_c := 'true';         // Parameter has op cont and mon capability
   S7_visible:='false';          // Parameter not visible
   S7_link := 'false'            // and cannot be linked
   }  :REAL := 0;         // Lower alarm limit (default 0)
...
  RUNUPCYC   {S7_visible:='false';
      S7_link:='false'} :INT := 3;     // Number of run up cycles
END_VAR
//*******************************************************************************
// Declaration Section: Static Variables
//*******************************************************************************
VAR
...
siRUNUPCNT   :INT := 0;     // Counter for RUNUPCYC execution
...
END_VAR
//*******************************************************************************
// Startup
//*******************************************************************************
  // OB100: Warm restart
  100:
...
  siRUNUPCNT := RUNUPCYC;   //  Save RUNUPCYC value
...
//*******************************************************************************
// Technological Section
//*******************************************************************************
  IF siRUNUPCNT = 0         // RUNUPCYC cycle already elapsed ?
  THEN
   IF (PV_IN > H_ALM) THEN  // If the process value violates the high alarm limit
     QH_ALM  := 1;          // set error output
     QL_ALM  := 0;          // reset error output
   ELSIF (LMN < L_ALM) THEN // If the process value violates the low alarm limit
     QL_ALM  := 1;          // set error output
     QH_ALM  := 0;          // reset error output
   ELSE
     QH_ALM  := 0;          // Reset error outputs
     QL_ALM  := 0;

   END_IF;
  ELSE
    siRUNUPCNT := siRUNUPCNT - 1;
  END_IF;
 END_CASE;
```

1.6.2 Suppressing Specific Messages

If you want to suppress specific messages that you expect to be otherwise generated, you can use the technique explained below:

Define an input parameter with the BOOL data type in your block that is evaluated in your block algorithm so that if the message is suppressed, the event is not passed to the SIG input of the ALARM block.

In the following example, the inputs M_SUP_AL and M_SUL_AH are used to suppress a single alarm:

```
//*****************************************************************************
// Declaration Section: Block Parameters
//*****************************************************************************

VAR_INPUT
.....
// Suppressing ALARM LOW
M_SUP_AL {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_string_0:= 'Suppress LL=No';   // Operator text for value (M_SUP_AL)= 0
    S7_string_1:= 'Suppress LL=Yes'   // Operator text for value (M_SUP_AL)= 1
      }           :BOOL               // Suppress alarm low

           // Suppressing ALARM HIGH
M_SUP_AH {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_string_0:= 'Suppress HH=No';   // Operator text for value (M_SUP_AH)= 0
    S7_string_1:= 'Suppress HH=Yes'   // Operator text for value (M_SUP_AH)= 1
      }           :BOOL;              // Suppress alarm high
...
END_VAR

//*****************************************************************************
// Messages with ALARM_8P
//*****************************************************************************
.....
 ALARM_8P_1(EN_R := TRUE,          // Update output ACK_STATE
   ID := 16#EEEE,                  // Data channel for messages (always 16#EEEE)
   EV_ID:= EV_ID,                  // Message number > 0
   SIG_1:= NOT M_SUP_AH AND QH_ALM, // Signal to be monitored 1 -> high alarm message
   SIG_2:= M_SUP_AL AND QL_ALM,    // Signal to be monitored 2 -> low alarm message
   SIG_3:= 0,                      // Signal to be monitored 3 -> no message
   SIG_4:= 0,                      // Signal to be monitored 4
.....
```

1.6.3 Compiling the Source File

When you have completed programming, you compile the source file using the SCL compiler. Select "File > Compile" or the click the Compile button in the toolbar. Following error-free compilation, the FB 501 block exists in the block folder of the S7 program.

For more detailed information, refer to the "S7-SCL for S7-300 and S7-400" manual.

1.7 Configuring Messages

General

If you want your block to send messages to the OS, you must define the multiple instance of an alarm block in the static variables.

With the ALARM_8 / ALARM_8P block, you can monitor up to 8 signals that you specify as parameters in the alarm block. Each time it is called, the block records the current state of the signals and sends a message to the OS the next time it is called if one of the signals has changed.

Configuring Messages in the SIMATIC Manager

You can edit EV_ID for the selected block in the SIMATIC Manager in the **Edit > Special Object Properties > Message** dialog.

You can interlock individual parts of the messages (for example, message text, message class etc.) to prevent changes; in other words, you can prevent the message from being modified in the block instance when you install your block in a CFC chart.

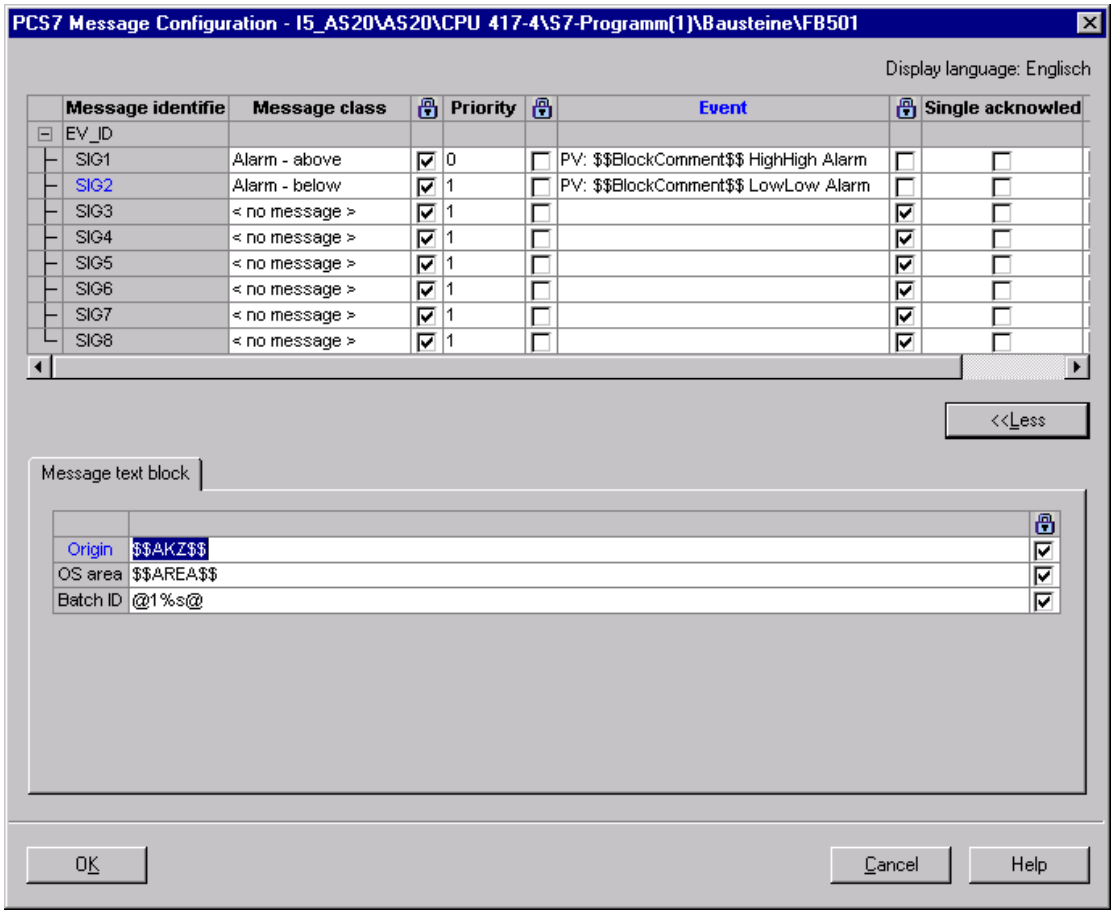


Figure 1-7 : Configuring Messages in the SIMATIC Manager

You first enter the texts for all the messages of this block. The individual texts correspond to the user text blocks in AlarmLogging in WinCC.

Origin:

Here, you can specify the origin of the message.

If you specify the keyword $$AKZ$$, the path of the hierarchy folder, the chart name, and block name are obtained and entered in the OS message texts when the data are transferred during the compilation OS.
Note: The PH path is entered only if the relevant hierarchy folders are included in the name (properties of the plant hierarchy folder or settings for the PH).

OS area:

Here, you can specify the area assignment of the message.

If you enter the keyword $$AREA$$ as the area or make no entry here, the corresponding attribute of the hierarchy folder is evaluated and entered in the OS message texts when the data are transferred during compilation of the OS.

Batch ID:

Here, you can specify a batch ID for the message.

If you enter the batch ID, the corresponding attribute is evaluated and entered in the "Charge Name" column in the message list of WinCC when the data are transferred during compilation of the OS. This is, however, not the batch ID but rather the batch name. If you want your block to be suitable for the optional S7 package SIMATIC BATCH, you must enter @1%s@ here. With this entry, the message includes the BATCH identifier as the first auxiliary value (see also Section 1.8). If you do not use SIMATIC BATCH, do not make any entry here.

Message Classes

You can then specify the message class for each message. As soon as you click on a message row in the message class column, the cell changes to a combo box in which you can select the message class. Messages that are not used must have the message class "< no message >". For more detailed information on handling messages, refer to the WinCC documentation.

Enter a description of the cause of the error in the "Event" column (maximum 40 characters including any auxiliary values) and whether or not the message must be acknowledged individually in the "Single acknowledgment" column (if the check box is selected) or whether the message can be acknowledged by a group acknowledgment.

In the "Locked" column (blue lock icon), you decide whether the message class can be modified instance-specifically in the CFC by the user of the block (check box not selected) or whether the message text is locked (check box selected).

Priority

Here you can assign different priorities to a message.

In the "Locked" column (blue lock icon), you decide to allow (check box not set) or lock (check box set) instance-specific modification of the priority in the CFC by the user of the block.

Event

Enter the message text in this field.

Auxiliary Values for Messages

If you want additional information (for example, measured values) to be transferred to the OS with the message, you must use an ALARM block that allows you to specify auxiliary values (ALARM_8P = 10 auxiliary values). The values transferred with the parameters SD_1 to SD_10 of the ALARM block can be included in the message texts as follows:

@ Parameternumber formatstatement @

In the example shown below, the value for the parameter SD_4 is displayed in decimal format. The format statements that you can specify comply with C syntax.

@4%d@

You can display the comment for the block instance in the message text by entering the keyword **$$BlockComment$$**.

In the "Locked" column (blue lock icon), you decide to allow (check box not selected) or lock (check box selected) instance-specific modification of the message text in the CFC by the user of the block.

Note: The auxiliary values of an ALARM_8P are of data type ANY, connector I/O. The permissible data types of the auxiliary values are listed in the ALARM_8P Online Help. If you define the auxiliary values as an input and not as an I/O in the block interface, you have to save the auxiliary values to a variable in the VAR section and transfer these variables as an auxiliary value with ALARM_8P.

The data type STRING may not be transferred as an auxiliary value. You have to implement it as a sample block, copy it to an ARRAY OF BYTE and transfer this ARRAY as an auxiliary value.

Single acknowledgment

Set the options box to enable individual acknowledgment for this message.

Infotext

Enter the info text in this field.

1.8 Linking SIMATIC BATCH

If you want to use your blocks with the "SIMATIC BATCH S7 optional package, define the following input and in/out parameters:

Parameter Name	Meaning	Parameter Type	Data type
BA_EN	BATCH enable	INPUT	BOOL
BA_NA	BATCH name	INPUT	STRING [32]
BA_ID	Consecutive batch number	INPUT	DWORD
OCCUPIED	BATCH occupied ID	INPUT	BOOL
STEP_NO	BATCH step number	INPUT	DWORD

Excerpt from the sample block:

```
//*******************************************************************************************
// Declaration Section: Block Parameters
//*******************************************************************************************
VAR_INPUT
....
                    // Parameters for SIMATIC BATCH
  STEP_NO {S7_visible := 'false';
     S7_m_c   := 'true'}  :DWORD;           // Batch step number

  BA_ID {S7_visible  := 'false';
     S7_m_c    := 'true'}   :DWORD;         // Batch ID

  BA_EN {S7_visible := 'false';             // Parameter not visible in CFC chart
     S7_m_c := 'true'                       // Parameter has op cont and mon capability
     }          :BOOL := 0;                 // Batch enable
  BA_NA {S7_visible := 'false';
     S7_m_c   := 'true'} :STRING[32] := ''; // Batch name


  OCCUPIED {S7_visible := 'false';
       S7_m_c   := 'true'} :BOOL := 0;      // Occupied by batch
....
END_VAR
```

To generate messages in a block such as the one shown above, you must use the inputs BA_NA, STEP_NO and BA_ID (in this order) as auxiliary values .

The auxiliary values have the following significance:

Associated Value	Meaning
1	BATCH name BA_NA
2	BATCH step number STEP_NO
3	BATCH: consecutive batch number BA_ID
4 to 7	Block-specific significance or free for the user

1.9 Creating CFC Block Types

In contrast to programming with SCL in which the variables are declared and assignments are programmed, CFC is based on the interconnection of graphic objects. This means that you can develop new blocks by positioning and interconnecting existing blocks.

The following description represents an overview and explains the basic procedures. For a detailed description of creating blocks in CFC, refer to the "CFC for S7" manual or the CFC online help.

1.9.1 Example: CONTROL2

The existing sample block "CONTROL" will be extended by adding a multiplier. The process value will be formed by multiplying two input values (IN1 and IN2). The extended block will be generated as CONTROL2 (FB 601).

To extend a block:

- Open a new CFC chart and place the sample block **CONTROL** in it.
- From the CFC library \ELEMENTA, drag a multiplier **MUL_R** (FC 63) to the chart.
- Interconnect the "OUT" output of **MUL_R** with the process value (parameter "PV_IN") of the sample block.
- In the chart, open the Interface Editor ("View" menu > chart I/Os) " or with the icon ) and select the icon "IN" in the left "Interface" window of the chart I/Os.
- Interconnect the inputs "IN1"and "IN2" of **MUL_R** with the chart I/Os; drag the block I/O to the chart I/O (right hand window).
- Switch the invisible I/Os - except EN and ENO - of the block CONTROL to visible (Object Properties > Tab: I/Os > Column: Invisible > reset set I/Os).
- Interconnect all visible I/Os (except for the process value that is already interconnected, as soon as EN and ENO) with the chart I/Os of the CFC chart.
- Compile the CFC chart as a block in the "*Chart > Compile > Chart as Block Type*" dialog.
  - Enter the FB number (in this case, 601) in the "General" tab. Then enter the other properties in the relevant fields: Symbolic name, Family, Author, Version. The name of the CFC chart has already been entered in the Name (header) field.
  - Enter the block attributes and system attributes you require in the "Attributes" tab. Do not specify the "S7_tasklist" system attribute here (refer to the following rule).
  - Start compilation with "OK".

Installation rule:

The CFC block type is installed in the default cyclic OB (e.g. OB 35) and in every OB in the task lists of the nested blocks; in other words, its task list is made up of the task lists of its nested blocks. The nested blocks themselves are only called in the OBs listed in their own task list. In the example shown here, this means the following:

- The sample block **CONTROL** has the task list "S7_tasklist = 'OB80,OB100' ".
- The multiplier **MUL_R** has no task list.
- The CFC block type therefore has the task list "S7_tasklist = 'OB80,OB100' ". Only **CONTROL** is called, however, in OB 80 and OB 100 but not **MUL_R**.

If the CONTROL block is not to be called in OB 35, you have to either delete it from OB 35 or move it to another OB. The same applies for the MUL_R block.

1.10 Naming Conventions and Numeric Range

Numeric Range

To prevent conflicts with the process control PCS 7 blocks supplied by Siemens, you should start numbering your blocks starting at number 501. When selecting the block numbers for your blocks, you should also keep in mind the performance data of the CPU types supported by your library.

Names

When naming your block parameters, you should keep to the following rule:

Binary outputs begin with **Q**, for example, QH_ALM or Q_L_ALM

1.11 Source Code of the Samples

```
//Author: ABC                       Date: 13.08.00         Vers.:1.00
//Modified:                         Date: 18.11.03         Vers.:
//Change:
//*****************************************************************************************
// Block header
//*****************************************************************************************

FUNCTION_BLOCK "CONTROL"
TITLE =        'CONTROL'
{  // List of system attributes
S7_tasklist:= 'OB80,OB100';   // Block called if a time error occurs and at warm restart
S7_m_c:=      'true';         // Block can be controlled and monitored
S7_alarm_ui:= '1';            // Setting for PCS 7 message dialog
}                             // ('0'=standard message dialog)
AUTHOR:          ABC
NAME:            CONTROL
VERSION:         '0.02'
FAMILY:          XYZ
KNOW_HOW_PROTECT

//*****************************************************************************************
// Declaration Section: Block Parameters
//*****************************************************************************************

VAR_INPUT
SAMPLE_T {S7_sampletime:= 'true';    // Param. of block sampling time (cycle of task)
       S7_visible:='false';          // Parameter not visible
       S7_link:= 'false'             // Parameter cannot be linked
      }          :REAL := 1;         // sample time [s] (default 1 sec)

  H_ALM {S7_m_c := 'true';
     S7_visible:='false';
     S7_link := 'false'}  :REAL :=100; // Upper alarm limit (default 100)

  L_ALM {S7_m_c := 'true';           // Parameter has op cont and mon capability
     S7_visible:='false';            // Parameter not visible
     S7_link := 'false'              // and cannot be linked
     }           :REAL := 0;         // Lower alarm limit (default 0)

  M_SUP_AL {S7_visible:='false';
     S7_link:='false';
     S7_m_c:='true';
     S7_string_0:= 'Suppress LL=No';  // Operator text for value (M_SUP_AL)= 0
     S7_string_1:= 'Suppress LL=Yes'  // Operator text for value (M_SUP_AL)= 1
      }            :BOOL;             // Suppress alarm low

  M_SUP_AH {S7_visible:='false';
     S7_link:='false';
     S7_m_c:='true';
     S7_string_0:= 'Suppress HH=No';
     S7_string_1:= 'Suppress HH=Yes'
      }            :BOOL;             // Suppress alarm high

  SP_OP_ON {S7_visible:='false';
     S7_dynamic:='true'   // CFC in test/commissioning: display of actual value in AS)
      }       :BOOL := 1; // Enable 1=operator for setpoint input

  SPBUMPON {S7_visible:='false';
     S7_link:='false';
     S7_m_c:='true';
     S7_string_0:='SP bumpless=Off';
     S7_string_1:='SP bumpless=On'
      }       :BOOL := 1; // Enable 1=bumpless for setpoint on
```

```
  SP_EXTON {S7_visible:='false';
     S7_dynamic:='true'    // CFC in test/commissioning: display of actual value in AS)
      }    :BOOL := 1;     // 1: Select SP_EXT

  SP_EXT   {S7_dynamic:='true'}

           :REAL := 0;                // External setpoint
  SP_HLM   {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_shortcut:='SP high limit';     // Text (max 16 chars) for display on OS
    S7_unit:=''}                      // Unit (max 16 chars)
            :REAL := 100;             // Setpoint high limit
  SP_LLM    {S7_visible:='false';
     S7_link:='false';
     S7_m_c:='true';
     S7_shortcut:='SP low limit';
     S7_unit:=''}
            :REAL := 0;              // Setpoint low limit

  PV_IN {S7_dynamic:='true';
     S7_m_c:='true';
     S7_unit:='%'}    :REAL := 0;    // Process value (to AUX_PR04 of message)

  GAIN   {S7_link:='false';
    S7_edit:='para';     // Parameter setting in Import/Export Assistant (IEA)
    S7_m_c:='true';
    S7_shortcut:='Gain';
    S7_unit:=''}      :REAL := 1;   // Proportional gain

EV_ID   {S7_visible:='false';
S7_link:='false';
    S7_param :='false';              // Parameter cannot be set in CFC
    S7_server:='alarm_archiv';       // Message no. assigned by server
    S7_a_type:='alarm_8p'            // Block signals with ALARM_8P
       }           :DWORD := 0;      // Message ID

                                    // Parameters for SIMATIC BATCH
STEP_NO {S7_visible := 'false';
      S7_m_c   := 'true'}  :DWORD;   // Batch step number
  BA_ID {S7_visible  := 'false';
      S7_m_c    := 'true'}  :DWORD;  // Batch ID
  BA_EN {S7_visible := 'false';      // Parameter not visible in CFC chart
      S7_m_c := 'true'               // Parameter has op cont and mon capability
     }             :BOOL := 0;       // Batch enable

  BA_NA {S7_visible := 'false';
      S7_m_c   := 'true'} :STRING[32] := '';  // Batch name
  OCCUPIED {S7_visible := 'false';
      S7_m_c   := 'true'} :BOOL := 0;         // Occupied by Batch

  RUNUPCYC   {S7_visible:='false';
      S7_link:='false'} :INT := 3;  // Number of run up cycles
  SUPPTIME       :REAL := 0;        // Sample delay
  SUPP_IN        :REAL := 0;        // Input value for sample delay
END_VAR
```

```
VAR_OUTPUT
  LMN {S7_shortcut:='pressure';     // Name of the parameter on OS
    S7_unit := 'mbar';              // Unit of the parameter
    S7_m_c := 'true'                // Can be monitored
    } :REAL;                        // Manipulated value

  QH_ALM  :BOOL := false;           // 1 = HH alarm active

  QL_ALM  :BOOL := false;           // 1 = LL alarm active

  QSP_HLM   {S7_visible:='false';
    S7_dynamic:='true'}   :BOOL := 0;       // 1=Setpoint output high limit active

  QSP_LLM   {S7_visible:='false';
    S7_dynamic:='true'}   :BOOL := 0;       // 1=Setpoint output low limit active

  Q_SP_OP   {S7_visible:='false';
    S7_dynamic:='true';
    S7_m_c:='true'}       :BOOL := 0;       // Status: 1=operator may enter setpoint

  QOP_ERR   {S7_visible:='false';
    S7_dynamic:='true'}   :BOOL := 0;       // 1=operator error

  QMSG_ERR   {S7_visible:='false';
    S7_dynamic:='true'} :  BOOL := 0;       // ALARM_8P: Error output

  MSG_STAT   {S7_visible:='false';
    S7_dynamic:='true'} :  WORD := 0;       // Message: STATUS output

  MSG_ACK   {S7_visible:='false';
    S7_dynamic:='true'} :  WORD := 0;       // Message: ACK_STATE output

  SUPP_OUT                :REAL := 0;       // Output value for sample delay
  SP   {S7_dynamic:='true';
    S7_m_c:='true'}       :REAL := 0;       // Active setpoint

END_VAR

VAR_IN_OUT
  SP_OP {S7_visible:='false';
    S7_link:='false';
    S7_m_c:='true';
    S7_shortcut:='Setpoint';
    S7_unit:=''}          :REAL := 0;      // Operator input setpoint

  // Freely assignable auxiliary values of ALARM_8P
  AUX_PR05 {S7_visible := 'false'} : ANY;  // Auxiliary value 5
  AUX_PR06 {S7_visible := 'false'} : ANY;  // Auxiliary value 6
  AUX_PR07 {S7_visible := 'false'} : ANY;  // Auxiliary value 7
  AUX_PR08 {S7_visible := 'false'} : ANY;  // Auxiliary value 8
  AUX_PR09 {S7_visible := 'false'} : ANY;  // Auxiliary value 9
  AUX_PR10 {S7_visible := 'false'} : ANY;  // Auxiliary value 10
END_VAR
```

```
//*****************************************************************************
// Declaration Section: Static Variables
//*****************************************************************************
VAR
  sbRESTART   :BOOL := TRUE;       // Initial start memory bit
  siRUNUPCNT  :INT := 0;           // Counter for RUNUPCYC execution
  sSUPP_IN    :REAL := 0;          // Old value of sample delay input
  ACT_TIME    :REAL := 0;          // Time counter

  srPV_IN     :REAL := 0;          // Auxiliary value for PV_IN
  swSTEP_NO   :DWORD;              // Batch step number
  sdBA_ID     :DWORD;              // Batch ID

  sbyBA_NA    :ARRAY[1..32] OF BYTE := 32(0);

//*****************************************************************************
// Declaration Section Multiple Instances
//*****************************************************************************
  OP_A_LIM_1:  OP_A_LIM;           // Operator control block 1
  ALARM_8P_1:  ALARM_8P;           // Generation of max. 8 messages with
                                   // max. 10 auxiliary values
END_VAR

//*****************************************************************************
// Declaration Section: Temporary Variables
//*****************************************************************************
VAR_TEMP
  // Start info: Structure with info for the OB that has just called the block
  TOP_SI:    STRUCT
    EV_CLASS :BYTE;
    EV_NUM   :BYTE;
    PRIORITY :BYTE;
    NUM      :BYTE;
    TYP2_3   :BYTE;
    TYP1     :BYTE;
    ZI1      :WORD;
    ZI2_3    :DWORD;
  END_STRUCT;

  // Start info: Structure with info for the last called startup OB
  START_UP_SI:  STRUCT
    EV_CLASS :BYTE;
    EV_NUM   :BYTE;
    PRIORITY :BYTE;
    NUM      :BYTE;
    TYP2_3   :BYTE;
    TYP1     :BYTE;
    ZI1      :WORD;
    ZI2_3    :DWORD;
  END_STRUCT;

  S7DT   :DATE_AND_TIME;           // Local time variable
  DUMMY  :INT;                     // Auxiliary variable
END_VAR
```

```
//*****************************************************************************
// Code Section
//*****************************************************************************

//*****************************************************************************
// Dependence on Calling OB
//*****************************************************************************
   // Read out start info with SFC6 (RD_SINFO)
   DUMMY := RD_SINFO (TOP_SI := TOP_SI, START_UP_SI := START_UP_SI);
 IF sbRESTART THEN
   // Initial start
   TOP_SI.NUM := 100;          // Execute initial start as warm restart
   sbRESTART := FALSE;         // Reset initial start
 END_IF;

   // In which OB was the block called ?
 CASE WORD_TO_INT(BYTE_TO_WORD(TOP_SI.NUM)) OF

//*****************************************************************************
// Handling Error OBs
//*****************************************************************************
// OB80: time error
    80:
    QH_ALM  := 0;               // Reset error outputs
    QL_ALM  := 0;
//*****************************************************************************
// Startup
//*****************************************************************************
// OB100: Warm restart
    100:
    QH_ALM  := 0;               // Reset error outputs
    QL_ALM  := 0;
    siRUNUPCNT := RUNUPCYC;     // Save RUNUPCYC value
ELSE
//*****************************************************************************
// Operator Input of Setpoint SP_OP (Real Value) or Connected Setpoint SP_EXT
//*****************************************************************************

   // Multiple instance call OP_A_LIM (for meaning of parameters,
   // see online help OP_A_LIM)
   OP_A_LIM_1(U := SP_OP, U_HL:= SP_HLM, U_LL:= SP_LLM, OP_EN:= SP_OP_ON, BTRACK:=
SPBUMPON, LINK_ON:= SP_EXTON, LINK_U:= SP_EXT);
   OK := OK AND ENO;          //Enter enable out of OP_A_LIM in OK flag of the block
   Q_SP_OP := OP_A_LIM_1.QOP_EN;     // 1: Enable operator input of SP
   QOP_ERR := OP_A_LIM_1.QOP_ERR;    // 1: Operator error
  QSP_HLM := OP_A_LIM_1.QVHL;        // 1: High limit
   QSP_LLM := OP_A_LIM_1.QVLL;       // 1: High limit
   SP   := OP_A_LIM_1.V;             // Effective setpoint

//*****************************************************************************
// Technological Section
//*****************************************************************************
IF (SUPP_IN <> sSUPP_IN) THEN
   ACT_TIME := SUPPTIME;           // Initialize time counter
   sSUPP_IN := SUPP_IN;
END_IF;
  IF (ACT_TIME > 0) THEN           // If wait time not yet elapsed
   ACT_TIME := ACT_TIME-SAMPLE_T;  // Count down wait time
  ELSE
   SUPP_OUT := SUPP_IN;            // Connect input to output
  END_IF;
```

```
  LMN := GAIN * (SP - PV_IN);       // Calculate manipulated variable

  IF siRUNUPCNT = 0                 // RUNUPCYC cycle already elapsed ?
  THEN
   IF (PV_IN > H_ALM) THEN          // If the process value violates
                                    // the high alarm limit
     QH_ALM  := 1;                  // set error output
     QL_ALM  := 0;                  / reset error output

   ELSIF (PV_IN < L_ALM) THEN // If the process value violates the low alarm limit
     QL_ALM  := 1;          // set error output
     QH_ALM  := 0;          // reset error output
   ELSE
     QH_ALM  := 0;          // Reset error outputs
     QL_ALM  := 0;

   END_IF;
  ELSE
    siRUNUPCNT := siRUNUPCNT - 1;
  END_IF;
END_CASE;

//*****************************************************************************
// Messages with ALARM_8P
//*****************************************************************************

  // STRING variables must not be linked to ALARM8_P as auxiliary values
  // so transfer in array of bytes
    FOR DUMMY := 1 TO 32
     DO
      sbyBA_NA[DUMMY] := 0;    // Delete array as default
     END_FOR;
     DUMMY := BLKMOV (SRCBLK:= BA_NA,DSTBLK:=sbyBA_NA);
     swSTEP_NO  := STEP_NO;    // Batch step number (due to I/O aux val ALARM_8P)
     sdBA_ID   := BA_ID;       // Batch ID     (due to I/O aux val ALARM_8P)
     srPV_IN   := PV_IN;       // Auxiliary value must not be a INPUT
 ALARM_8P_1(EN_R := TRUE,            // Update output ACK_STATE
      ID := 16#EEEE,                 // Data channel for messages (always 16#EEEE)
      EV_ID:= EV_ID,                 // Message number > 0
SIG_1:= NOT M_SUP_AH AND QH_ALM,     // Signal to be monitored 1 -> high alarm message
SIG_2:= NOT M_SUP_AL AND QL_ALM,     // Signal to be monitored 2 -> low alarm message
      SIG_3:= 0,                     // Signal to be monitored 3 -> no message
      SIG_4:= 0,                     // Signal to be monitored 4
      SIG_5:= 0,                     // Signal to be monitored 5
      SIG_6:= 0,                     // Signal to be monitored 6
      SIG_7:= 0,                     // Signal to be monitored 7
      SIG_8:= 0,                     // Signal to be monitored 8
      SD_1 := sbyBA_NA,              // Auxiliary value 1
      SD_2 := swSTEP_NO,             // Auxiliary value 2
      SD_3 := sdBA_ID,               // Auxiliary value 3
      SD_4 := srPV_IN,               // Auxiliary value 4
      SD_5 := AUX_PR05,              // Auxiliary value 5
      SD_6 := AUX_PR06,              // Auxiliary value 6
      SD_7 := AUX_PR07,              // Auxiliary value 7
      SD_8 := AUX_PR08,              // Auxiliary value 8
      SD_9 := AUX_PR09,              // Auxiliary value 9
      SD_10:= AUX_PR10);             // Auxiliary value 10

    QMSG_ERR := ALARM_8P_1.ERROR;    // ERROR status parameter
    MSG_STAT := ALARM_8P_1.STATUS;   // STATUS status parameter
    MSG_ACK := ALARM_8P_1.ACK_STATE; // Current OS confirmation status
END_FUNCTION_BLOCK
```

2 Creating Faceplates

2.1 General Notes on the Configuration

The chapters below provide you with all the information you need to create a faceplate for PCS 7. This covers the use of the WinCC tools you require to create the faceplates and details on how to work with the Faceplate Designer.

Note

The block icons and faceplates shown in this description are examples and may deviate in minor details from the display of the actual objects.

Requirements and Previous Experience

The faceplates described in this documentation are designed for use in WinCC. To create the blocks, you require the basic WinCC package, including the process control options "Basic Process Control" and "Advanced Process Control".

It is assumed that you have participated in the following courses:

- SIMATIC WinCC System Course
  (offered by the A&D Training Center under ST-BWINCCS)
- SIMATIC WinCC Openness N
  (offered by the A&D Training Center under ST-BWINCCN)

2.1.1 Steps in Creating a Faceplate

Procedure

The following steps have proved to be the most successful method of creating a faceplate:

- Designing the faceplate
- Configuring the faceplate
- Testing the faceplate

2.1.1.1 Designing the Faceplate

Visualization

The faceplate represents the operator control and monitoring interface to an AS block. There are two ways in which a faceplate can be displayed:

- **Group display:** Display of the AS values in various views, with an element for selecting the loop display.
- **Loop display:** Display of the elements of all views of the group display.

System Attributes

The input, output, and in/out parameters of an AS block that can be controlled and monitored is specified in the system attributes when the AS block is created. For details of the system attributes, refer to the description of the declaration section in "Structure of an AS Block" Table 1-1 and Table 1-2.

Parameters

The parameters are selected according to the following criteria:

- Which data do the operators require to recognize the current status quickly and without the risk of misinterpretation?
- How will these values be displayed?
- Which values will operators be able to change?
- Which authorization level is required for the particular operator input?
- Are process-dependent operator input interlocks necessary?
- In which view of the faceplate will the individual values be displayed?

  Tip: Group the individual parameters according to their functions. Place the most important elements and particularly those that change continuously in the "Standard" view.

Design

After the parameters are specified and displayed, the faceplate is designed; in other words, the picture elements, their names, the parameters setting of the picture elements, and their positions are selected. Always use names that relate directly to the displayed object and that can also be pronounced easily.

Example: You want to display the "OCCUPIED" variable in a status display → name the status display "OCCUPIED".
Keeping to this procedure makes both project documentation and maintenance easier.

2.1.1.2 Configuring the Faceplate

You can use the "WinCC Graphics Designer" tool to configure the faceplate. Starting with the templates provided by the Faceplate Designer, you convert your picture design into WinCC pictures. For a detailed description, refer to Section 2.1.2. "Creating Faceplates with the Faceplate Designer".

2.1.1.3 Testing the Faceplate

You should test the faceplate in two steps:

1. Check the properties of the pictures you have created in WinCC Explorer:
   - Are the parameter names written correctly?
   - Do parameters that are displayed more than once have the identical WinCC cycle? Different cycles can confuse operators (for example, inconsistency between the bar graph display and the numeric display) and increase the communication load on the system.
   - Are the direct connections correct?
   - Are all the event-triggered scripts available?
2. Check in WinCC Runtime:
   - Does the faceplate open in the group display when you click on the block icon?
   - Can you change between individual views in the group display?
   - Does the faceplate open in the loop display when you click the "loop display" button?
   - Are the values of the AS block displayed correctly?
   - Is the display of the message and trend view correct?
   - Does the enabling of the controllable parameters function correctly?
   - Are the values written to the AS block following operator input?
   - Is the operator input log correct?

2.1.2 Creating Faceplates with the Faceplate Designer

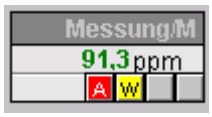
One of the components of the "Advanced Process Control" WinCC optional package is the Faceplate Designer. This tool generates the templates for PCS 7-compliant creation of faceplates.

2.1.2.1 Templates of the Faceplate Designer

WinCC provides the following templates for the creation of faceplates:

- Block icons (preconfigured icons for process pictures)
- Template pictures
- Object construction kit with objects for creating faceplates
- Global scripts

2.1.2.2 Block Icon Templates



The block icons can be found in the WinCC picture "@@PCS7Typicals.PDL", for example Valve, Drive, Measured Value, Controller etc.

- The sample templates can be edited and modified so that you can change the shape, color, layout etc. and adapt them to faceplates created for a specific project.
- The ready-made call scripts for the faceplates are already included and do not need to be configured.
- They can be interconnected quickly and simply using the "Connect picture block to tag structure" dynamic wizard.

  For detailed description refer to Chapter 2.8, "Block Icons".

Note:

If you make changes, save the picture under the name "@PCS7Typicals.PDL". The search engine in the PH searches for these names first. The templates from "@@PCS7Typicals.PDL" are used only if these are not found.

2.1.2.3 Template Pictures

The pictures and bitmaps are stored in the "WinCC\options\pdl\FaceplateDesigner" folder and are copied to the project using the OS project editor. For more information, refer to the Online Help "OS Project Editor Basic Data“.

Object Construction Kit

The WinCC picture "@PCS7Elements.PDL" contains a selection of ready-made objects (user objects) for creating a faceplate, for example, I/O fields, texts etc. The picture is stored in the path
"Siemens\WinCC\options\pdl\FaceplateDesigner_V6"
and copied to the "GraCS" folder of the project directory during a run of the OS Project Editor.

For details, refer to Chapter 2.3, "Basic Elements".

Global Scripts

The faceplate calls take the form of global scripts and are stored in the folder "WinCC\aplib\FaceplateDesigner".

For details, refer to Chapter 2.4, "Scripts".

2.1.2.4 Configuration Steps

Notes on Configuration

- The faceplates created with the Faceplate Designer are initially stored in the "GraCS“ directory of the currently opened project.
  The directory "\Siemens\WINCC\options\pdl\FaceplateDesigner“ is provided for storing user faceplates for the projects..

  In the OS project editor, you can select the file segments to decide which of the default faceplates from the folder
  "\Siemens\WINCC\options\pdl\FaceplateDesigner_V6“
  and which of the user faceplates from the
  "\Siemens\WINCC\options\pdl\FaceplateDesigner“
  directory need to be copied to the project when you generate the basic data.
- The data from the "\Siemens\WINCC\options\pdl\FaceplateDesigner“also need to be copied to the corresponding directory of the WinCC clients.

If required, you can read/write protect the functions you have configured in your own blocks in the "Global Script" editor.

For more information, refer to the documentation on the "Global Script" editor.

- The dynamics of the faceplates created with the Faceplate Designer can be controlled completely in your configuration. The performance of a faceplate therefore depends in particular on the selection of suitable dynamics during configuration. In this respect, an optimized, "trim" interface between the AS and OS functions is particularly important. This applies above all to the block icons.

  Note
  It is here advisable to use the status word "VSTATUS" (particularly for the block icons), which has been introduced in V6 for the AS blocks, in order to reduce the number of variable links.
  This new status word, however, could not be used in this case, since the standard block icons must also be capable of visualizing AS blocks of the previous V5.

- The faceplate dynamics may not contain any C script that defines fixed coded instance names, since these are type-specific.

2.1.3 Operator authorization

You can assign authorizations at the faceplates to grant access to specific instances.

The configuration or instance-specific information is stored at the block icons.

At the two block icon properties "Processcontrolling_backup" and "Higherprocesscontrolling_backup", you can set the authorization level for process controlling and for higher-level process controlling.

The information is passed to the faceplate by means of a script.

These properties are set by default to the usual authorization levels 5 and 6, and may be modified to suit requirements. If you do so, note that the levels 1 to 10 are assigned a fixed meaning in WinCC.

The parameters which are allocated to the two authorization levels are derived from the previously used faceplates. Process controls such as "On", "Off", "Manual", "Auto", "Set setpoint " are assigned to authorization level 5, whereas higher-priority parameters such as "Limits", "Control Parameters" etc. are assigned to level 6.

The area-specific user authorizations remain untouched.

Area-specific authorizations are verified on the basis of the assigned variable name (tag name). For this, however, it is necessary for the corresponding area name to be configured at the "OS area" parameter in the PH and to be consistent with the actual OS area in the Picture Tree.

2.1.3.1 Configuring Authorizations for Basic Elements

The objects "@Level5" and "@Level6" for the configuration of authorizations are available in the views created in the Faceplate Designer.

New basic element created in a view can be directly linked to the objects "@Level5" and "@Level6" by means of direct connection.

These objects may not be deleted, since they are write accessed by the scripts.

The "Background Color" and "Operator control enabling" properties are linked, in order to gray out the field if an authorization does not exist.

If the operator control enabling is required for the access to objects of AS parameters, the password level for the authorization can be interconnected by means of direct connection. In this case, however, instead of grayed out objects, the message "No Authorization" is displayed if the authorization does not exist.

A more suitable means of linking WinCC authorizations and process-dependent authorizations to a control element of the basic elements is provided by the "Permission" object. The configuration of this object is described at this location.

The values of these properties are controlled by means of the "PCS7_UpdatePermission_V6" script.

By default, the "@Level5" object holds authorization level 5, and "@Level6" the authorization level 6. These values are provided by the block icon by means of the "Processcontrolling_backup" and "Higherprocesscontrolling_backup" properties and the specific instances can thus be modified.

Procedure:

1. Select the object for the authorization level (e.g. @Level6) and select "Object properties".
2. In the "Event → Property Topics → Colors → Background Color" tab, select direct connection.
3. Under "Target" (window on the right) and "Object in Picture", select the new object and the background color property.
4. Select a direct connection from the "Event → Property Topics → Other → Operator control enable" tab. For this event, a script is already available which at the same time, controls the background color if the operator control enabling is controlled by an AS variable; this script must then be deleted.
5. Under "Target" (window on the right) and "Object in Picture" select the new object, then select the operator control enable property.

To add objects to a view, select the last object interconnected by means of direct connection and repeat the procedure described above.

Important!

The configuration of the "Background Color" and "Operator control enable" properties must conform to the setting in "@Level5" or "@Level6" (e.g. "gray" for the 'background color' and "false" for the operator control enable) to ensure correct execution of the scripts.

The background is here:

If the script writes a value that already exists in the configuration of the property, this value is not passed to other properties by means of direct connection. You must therefore ensure that the configuration values of the source properties and of the target properties receiving the values via a direct connection are identical.

2.1.4 Modifying the Overview

In order to improve performance, you now need to remove some of the elements and scripts again from the "@PG_XXXXX_Overview.Pdl" picture when you create a faceplate without messages or group display. Remove the following items:

- Button16, message acknowledgment
- Button17, disable/enable messages
- @Level5, authorization
- MSG_LOCK, icon for message suppression
- In the picture object itself, for the "Picture Selection" event, the script.
- The batch icon "OCCUPIED“ also if the batch parameter "Occupied" does not exist.

In the Faceplate Designer you can take this into account by means of the check boxes "No group display" or "No batch parameters".

2.1.5 Configuring a Multiple Instance

Multiple instance means that several blocks that can be controlled and monitored in a faceplate on the OS are configured in a CFC chart.

During configuration, the icon is always interconnected with a full variable name.

Authorizations are verified not only for the selected variable name, but for the entire faceplate.

In the pictures "@PG_xxxxx.pdl" and "@PL_xxxxx.pdl", you must set the MULTI_INSTANCE property to 'TRUE' at the "@Faceplate" object, so that the "multiple instance" type is recognized and the block name for the various views is truncated from the variable name after the authorization has been verified.

All variable links in the faceplate must be appended the "/block name“.

e.g. **/Controller.PV_IN.**

Important: This must be included in the scripts!

All interconnections must be complemented accordingly in the OVERVIEW picture.

At the "@Faceplate" object in the picture "@PG_xxxxx.pdl" and "@PL_xxxxx.pdl", assign the properties of the batch parameters for the operator log to a block. Multiple-instance techniques are here not taken into consideration. To cover special situations, e.g. different BatchIDs, StepNo. etc. for the various blocks, this must be controlled by scripts during the selection of pictures in the various views.

In the trend view, the multiple instance technology is not taken into account in "Mode = 2“.

Online trends can only be displayed for the tag name stored in the block icon. If you want to display trends for several blocks, you can do this only using the other modes.

To implement multiple message blocks, the following items must also be observed and configured in the OVERVIEW picture:

- A sufficient number of group displays must be installed to match the number of installed message blocks.
- The script for the message acknowledgment button must be extended.
  Script "Button16/Mouseclick“.
  With the acknowledgment, the function "CSigAPIAcknowledgeTagAndCreateLTM“ must be executed for each instance.

  Example:

```
TCHAR sztag1[_MAX_PATH + 1] = "";
TCHAR sztag2[_MAX_PATH + 1] = "";

GetComputerNameA(szStation, &dwSize);
pszParentPicture = GetParentPicture(lpszPictureName);
lpszCurrentUser  = GetPropChar(pszParentPicture ,"@Faceplate","CurrentUser");
pszTagname = GetPropChar(pszParentPicture ,"OverviewWindow","TagPrefix");

strcpy(sztag1,pszTagname);
strcpy(sztag2,pszTagname);
strcat(sztag1,"/Instanz1");
strcat(sztag2,"/Instanz2");
printf ("sztag2 : %s\r\n", sztag2);

if (!CSigAPIAcknowledgeTagAndCreateLTM(sztag1, lpszCurrentUser , szStation, &Err))
      printf ("Error at CSigAPIAcknowledgePicture : %s\r\n", Err.szErrorText);
if (!CSigAPIAcknowledgeTagAndCreateLTM(sztag2, lpszCurrentUser , szStation, &Err))
      printf ("Error at CSigAPIAcknowledgePicture : %s\r\n", Err.szErrorText);
```

- The script for the “Disable /enable messages“ button must be extended.
  Script "Button17/Mouseclick“.
  With “Disable/enable messages”, the function
  “CSigAPILockMessage“ must be executed for all instances.

  Example:

```
DWORD dEventState1 =  GetPropWord(lpszPictureName,"EventState","CollectValue");
DWORD dEventState2 =  GetPropWord(lpszPictureName,"EventState2","CollectValue");
TCHAR sztag1[_MAX_PATH + 1] = "";
TCHAR sztag2[_MAX_PATH + 1] = "";

pszParentPicture = GetParentPicture(lpszPictureName);
pszTagname = GetPropChar(pszParentPicture ,"OverviewWindow","TagPrefix");
lpszCurrentUser  = GetPropChar(pszParentPicture ,"@Faceplate","CurrentUser");
GetComputerNameA(szStation, &dwSize);

strcpy(sztag1,pszTagname);
strcpy(sztag2,pszTagname);
strcat(sztag1,"/R");
strcat(sztag2,"/M");

if ((dEventState & 65536) == 65536)
{
    bRet =  CSigAPILockMessage(FALSE, sztag1, lpszCurrentUser  ,  szStation, &Err);
    SetPictureUp(lpszPictureName,lpszObjectName,"@Lock.bmp");
}
else
{
   bRet =  CSigAPILockMessage(TRUE, sztag1, lpszCurrentUser  ,  szStation, &Err);
   SetPictureUp(lpszPictureName,lpszObjectName,"@Unlock.bmp");
}

if ((dEventState2 & 65536) == 65536)
{
    bRet =  CSigAPILockMessage(FALSE, sztag2, lpszCurrentUser  ,  szStation, &Err);
    SetPictureUp(lpszPictureName,lpszObjectName,"@Lock.bmp");
}
else
{
   bRet =  CSigAPILockMessage(TRUE, sztag2, lpszCurrentUser  ,  szStation, &Err);
   SetPictureUp(lpszPictureName,lpszObjectName,"@Unlock.bmp");
}
```

- A message suppression icon must be installed for all message blocks (MSG_LOCK).
- An OCCUPIED icon must be installed for all batch-relevant blocks.

2.1.6 Configuring Number Formats

The relevant block icons are assigned three new number formatting properties :

Format_InputValue	For input values at the block icon, such as "Setpoint" and "ProcessValue"
Format_OutputValue	For the manipulated variable at the block icon
Format_xx	For any other values

Note: The selected terms "Format_InputValue“ and "Format_OutputValue“ are process-oriented and do not have a direct reference to the I/Os of the AS block parameters.

Example: At the FMT_PID, the parameter "PV“ forms an output parameter at the AS block, and in the process view it forms an input variable.

Format_InputValue and **Format_OutputValue** set the analog value formats of the block icon.

The default setting of **Format_InputValue** shows floating-point numbers with two decimals; at least one number (0.##) is displayed

The default setting of **Format_OutputValue** shows two floating-point numbers with two decimals; at least one number (0.##) is displayed.

The project engineer can adapt these default settings to suit a specific instance.

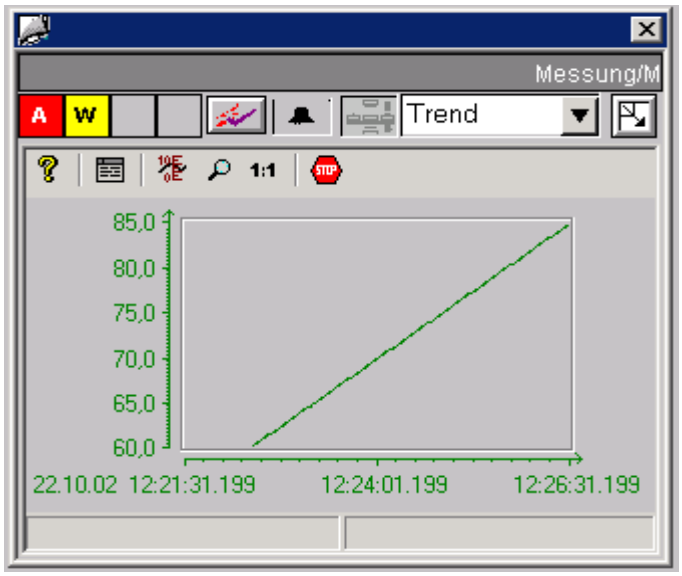
All three format properties of the block icons are transferred to the faceplate by means of a script, and can there be assigned to the analog values by means of direct connection.

Details:
The format properties of the block icon are transferred and saved to the “Format” object of the master prototype picture "@PG_xxxxx.pdl" by means of the "PCS7_OpenGroupDisplay_V6" script.

If a view contains a "Format“ object that is assigned the "Format_InputValue, Format_OutputValue, Format_xx" properties, these formats are transferred directly to this object by means of the "PCS7_Format_V6" script when a picture is selected.

From this object you can now assign any of these three formats to the analog values in this view by means of direct connection.

2.1.7 Configuring the Trend View



New properties are now available for the creation of trend pictures at the block icons for controllers, measurements etc.:

- **ReturnPath:** Here, the trend data for the corresponding process tag are transferred.
- **StandardTrend:** Determines which trend functionality is displayed in the trend view.

Standard-trend:	Mode
0	Trend functionality as in V5; a separate trend picture must be generated for each instance. The trend view "@PCS7_TREND.pdl" is available for all PCS7 faceplates and contains the "TrendPicture" window. To show a trend picture for a process tag at this point, proceed as follows: 1. Open the WinCC Editor "Tag Logging". 2. Create a variable archive for the trend view by means of the Archive Wizard. 3. Create the variables for this archive. For detailed instructions on the previous steps, refer to the WinCC Online Help, index "Tag Logging". 4. In the Graphics Designer, open the picture "@CONL1_Standard.pdl" 5. Configure the TlgOnlineTrend object "Control1" to suit your requirements; use the "Help" button to call further information. 6. Save the picture under the name "@CONL1_<tagname>.pdl". The variable name >tagname> and the process tag name of the AS block must be identical. Replace the "/" in the name with a "_" character, since Windows rejects filenames which contain a slash "/". Example of the window name of a process tag with the variable name "Measurement/M“ @CONL1_Measurement_M.pdl

Standard-trend:	**Mode**
2	A picture "@CONL1_Standard.pdl“ is called, in which one or several online variables are entered. An archive is not required. x Axis = 5 min This default setting avoids configuration effort and allows you to ignore naming conventions, while still being able to display the trends. The **ReturnPath** property in the block icon returns the number of trends, the name of the structured element as well as the color.
>2	A picture "@CONL1_Standard.pdl“ is called, in which one or several online archive variable names are entered. The following conventions must be observed for this configuration to work without requiring further effort: • The archive name is "Prozesswertarchiv“ (this is the default when you create the first archive). • The archive must be placed on the same server as the actual variables. • The archive variable name must conform to the default name assigned when the archive variable is created. • If the server prefix is contained in the "tagname" property of the block symbol, the server prefix has to be specified without "::" in the archive variable name. • If there is no server prefix entered in the "tagname" property of the entered variable name, the archive variable name has to be named exactly the same as the variable name. **Note:** The archive variable name in the older versions does not contain the separators "/" in "tagname". As of version V6, this is the "tagname" which must always be assigned to the archive variable in order to ensure correct functioning of the trend in the faceplate. The **ReturnPath** property of the block icon returns the number of trends, the variable name as well as the color. The duration [min] of the X axis for this mode is read from the "StandardTrend“ parameter. **Syntax for ReturnPath:** .PV_IN Name of the structured element, incl. the point for the first trend : Separator between structured element name and the color CO_GREEN Color of the first trend , Comma for further trends Example: .PV_IN:CO_DKGREEN,.SP:CO_BLUE,.LMNR_IN:CO_DKRED (Default for CTRL_PID)

Standard-trend:	**Mode**
	The sample shows three controller trends (process value, setpoint, feedback of the manipulated variable). If the four conventions described above cannot be maintained, the "ReturnPath“ property can be assigned additional parameters: ***tagname:** Custom archive tag name ***archivname:** Custom archive name (entered without “\“) ***archivserver:** Curve is located on an archive server, (enter server prefix of archive server without “::“) **Example:** .PV_IN:CO_DKGREEN,.SP:CO_BLUE,.LMNR_IN:CO_DKRED*tagname: MySpecialTag*archivname:MySpecialArchivname*archivserver: MySpecialArchivServer **Note:** Here, only a “tagname“, "archivname“ or "archivserver“ can be specified at the end of the string. In the case of multiple instances, the archive tag names must be adapted accordingly. The archive tags may only differ in the member tags. **Example of a multiple instance:** A CFC chart called "MULTI“ contains two MEAS_MON blocks "M1“ and "M2“. As default, the archive tag names for the measured value "U“ are generated as "MULTI/M1.U“ and "MULTI/M2.U“. The archive tags must be renamed to "MULTI.M1_U“ and "MULTI.M2_U“ Text string in the **ReturnPath** property in the block icon**:** .M1_U:CO_DKGREEN,.M2_U:CO_BLUE*tagname:MULTI*archivname:SystemArchive ***asia:** Add server prefix to archive tag If the *asia: parameter is not used, for a "tagname" with a server prefix it is also included with the derived archive tag name. If there is no server prefix in "tagname", no server prefix is used in deriving the archive tag name. ***asia: MyServerPrefix** The archive tag name contains the server prefix without "::“, but the "tagname“ property of the block symbol does not. In this case the server prefix can be supplemented (entry without "::“). ***asia:** The archive tag name contains no server prefix. In this case, the server prefix can be hidden. This variant should also be used when no server prefix is entered in "tagname" on the server since the server prefix is always available when the picture of the client is called. Example for *asia: display server prefix: .U:CO_DKGREEN*asia:OS(1)_PC2 Example for *asia: hide server prefix .U:CO_DKGREEN*asia:

The following table lists the required supplemental parameter settings in relation to:

- The form of the archive tag name and tag names in the block symbol
- From where the faceplate is called
- The archive tag on the local archive server

	Server	**Client with server picture**	**Client with client picture**	**Client with server picture and archive tag on separate archive server**	**Client with client picture and archive tag on separate archive server**
Tag name without server prefix archive tag name without server prefix	*asia:	*asia:	Not possible, server prefix is always in the "tagname"	*asia: *archivname: MyArchivserver prefix	Not possible, server prefix is always in the "tagname"
Tag name with server prefix archive tag name without server prefix	*asia:	*asia:	*asia:	*asia: *archivname: MyArchivserver prefix	*asia: *archivname: MyArchivserver prefix
Tag name without server prefix archive tag name with server prefix	*asia: MyServer prefix	*asia: MyServer prefix	Not possible, server prefix is always in the "tagname"	*archivname: MyArchivserver prefix	Not possible, server prefix is always in the "tagname"
Tag name with server prefix archive tag name with server prefix	No setting required with recommended tag settings	No setting required with recommended tag settings	No setting required with recommended tag settings	*archivname: MyArchivserver prefix	*archivname: MyArchivserver prefix

The block icon transfers the value of the "StandardTrend“ and "ReturnPath“ properties to the faceplate when the prototype picture "@PG_xxx“ is called.

The prototype picture contains a "Trend functions“ object with the "StandardTrend“ and "ReturnPath“ properties. The information is saved to this object and is processed by the "PCS7_Trend.fct" script when the "@PCS7_Trend.pdl" picture is called.

The same functions are also transferred to the loop display.

2.1.8 Configuring Different Block Icons and Faceplate Types for an AS Block Type

You can create a variant of a block icon for an AS block simply by changing the "type" properties of this icon.

To create a variant of a faceplate for an AS block, you assign it a name that is different to the AS block type name. A variant for "MEAS_MON“, for example could be the faceplate name "MEAS_NEU“.

The following actions are required at the block icon:

1. Rename the "Servername“ property in "PCS7 MEAS_NEU Control“.
2. Create the new "StructureType“ property at the block icon that contains the AS block type ("MEAS_MON“). This is necessary to ensure that the "Link faceplate to process tag“ wizard is still able to select the variable.

2.1.9 Web Client (Differences Compared with WinCC)

Below you will find the differences between WinCC and Web Client that must be taken into account.

2.1.9.1 Picture Names

In WinCC scripts, the picture names are addressed in absolute format, for example:

@screen.@win12:@1001.@top09:@pg_elap_cnt.OverviewWindow:@PG_ELAP_CNT_OverView.pdl

or in relative format, for example

@PG_ELAP_CNT_OverView.pdl

With a Web client, only the relative addressing works starting from the context in which the script runs.

When processing the Web client script, the picture window name and not the picture name must be specified as the address.

Example:
SetPropChar(lpszPictureName,lpszObjectName,lpszPropertyName,szValue);

WinCC: the lpszPictureName is a pointer to the picture name

Web client: the lpszPictureName is a pointer to the picture window name

An exception to this is the picture to which the script belongs and the parent picture; here, the picture name can be used as the pointer. It is, however, always advisable to use the picture window name as the pointer since this is unique, while the picture name can occur more than once.

Example of relative picture addressing:

In a picture @PG_xxx.pdl, there is a picture window "View“ in which, for example, the standard view of a faceplate is displayed.
If a different picture window needs to be addressed in the picture @PG_xxx.pdl from the picture (standard view) loaded in the "View“ picture window, for example "OperationWindow“ to open an operator box, the relative picture window address is obtained as follows:

sprintf(szPictureName,"../OperationWindow");

2.1.9.2 Upper-/Lower Case in File Names

File names (for example with pointers to picture names), are generally supplied in WinCC in upper case letters, for example @PG_MEAS_MON.PDL.

On the Web client, the file names are supplied exactly as they were created, for example, @Pg_Meas_Mon.pdl.

When comparing strings, it may be necessary to make adaptations here.

Example:
(strcmp(GetPictureUp(lpszPictureName,lpszObjectName),
"@KEEPVISIBLEON.EMF") == 0)

This always works in WinCC, however only on the Web if the file name was created only using upper case letters.

Necessary adaptation:
(strcmp(_**strlwr**(GetPictureUp(lpszPictureName,lpszObjectName)),
"**@keepvisibleon.emf**") == 0)

2.1.9.3 Loading Pictures in Picture Windows

In WinCC, after loading a picture in a picture window (for example with SetVisible), you can address the objects it contains directly with a script and change the properties.

On the Web client, pictures are loaded asynchronously. As a result, after loading a picture, the subsequent script processing must be delayed if objects of the picture are addressed in the same script.
To allow this, the function
WaitForDocumentReady("picture window name"); is available for the Web client.
The function delays the running of the script until the picture window name specified as the parameter reports back that the picture is loaded.

2.1.9.4 Deselecting Pictures within Scripts

In WinCC, "set invisible“ can be executed for a picture from within a script running in the context of this picture at any point in the script since the script processing task continues to process the task regardless of whether or not the picture is closed.

On the Web client, the "set invisible“ must be the last action of the script to execute.
Script commands following "set invisible“ are no longer executed because the script terminates at this point.

2.1.9.5 Difference in the Runtime Environment WinCC <-> Web

In the scripts of graphics or project functions, it is necessary to be able to distinguish the runtime environment, in other words, whether the script is running in WinCC or on the Web client.
For this purpose, there is the compiler instruction #ifdef RUN_ON_WEBNAVIGATOR or the negation #ifndef RUN_ON_WEBNAVIGATOR

This allows the differences between WinCC and the Web client to be taken into account, for example, script delay with WaitForDocumentReady, differences in picture addressing, different function names in process control functions, an functions that are not supported on the Web.

Example from the PCS7_ChangeView script:

```
#ifdef RUN_ON_WEBNAVIGATOR
        SetPropChar("../", "View", "PictureName", szViewName);
        WaitForDocumentReady("../View");
#else
        SetPropChar(lpszParent, "View", "PictureName", szViewName);
#endif
```

Note:

The syntax of the code section for Web is not checked when the WinCC script is compiled, but only when the pictures are published.

2.1.9.6 Function Names in WinCC / Web

Functions that trigger an action were mapped to other function names for the Web so that the action is triggered in the Web navigator client and not in WinCC runtime.

Example:

SSMRTChangeWorkdfield changes the working window in WinCC runtime and

SSMChangeWorkdfield changes the working window in the Web client.

You will find a list of supported functions and their names in the appropriate product descriptions. For the "Faceplate Designer" product, the following functions are supported:

void PCS7_ChangeView(char* lpszPictureName, char* lpszObjectName);

BOOL PCS7_CheckPermission(char* lpszTagname, DWORD dwLevel);

void PCS7_UpdateGroupPermission(char* lpszPictureName);

void PCS7_UpdateGroupTagname(char* lpszPictureName, char* lpszObjectName, char* value);

void PCS7_UpdateLoopTagname(char* lpszPictureName, char* lpszObjectName, char* value);

void PCS7_UpdatePermission_V6(char* lpszParentPictureName, int CallFrom, char* lpszPictureName);

void PCS7_OpenGroupDisplay_V6(char *lpszPictureName, char *lpszObjectName);

void PCS7_OpenLoopDisplay_V6(char* lpszPictureName);

void PCS7_OpenInputBoxBin_V6(char *lpszPictureName,char *lpszObjectName, int CallFrom);

void PCS7_1vnStati_Variable_Changed_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, double value);

void PCS7_OperationLog_V6(char* lpszPictureName, double dOldValue, double dNewValue, char* lpszOperationText, char* lpszUnit);

void PCS7_Trend_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, char * ArchivVar, char* OnlineVar, DWORD TrendColor, int CallFrom);

void PCS7_UpdateGroupTagname_V6(char* lpszPictureName, char* lpszObjectName, char* value);

void PCS7_UpdateLoopTagname_V6(char* lpszPictureName, char* lpszObjectName, char* lpszTagName);

void PCS7_UpdateBarLimits_V6(char* lpszPictureName, char* lpszObjectName, int CallFrom);

void PCS7_UpdateBar_V6(char* lpszPictureName, char* lpszObjectName, int CallFrom);

void PCS7_OpenInputBoxAnalog_V6(char *lpszPictureName,char *lpszObjectName,int CallFrom);

void PCS7_OpenGroupDisplay_I_V6(char *lpszPictureName, char *lpszObjectName, char*lpszInterlokName);

void PCS7_OpenComboBox_V6(char *lpszPictureName,char *lpszObjectName,int CallFrom);

void PCS7_OpenCheckBox_V6(char *lpszPictureName,char *lpszObjectName, int CallFrom);

void PCS7_Open3ComboBox_V6(char *lpszPictureName,char *lpszObjectName,int CallFrom);

void PCS7_Format_V6(char* lpszPictureName, char* lpszObjectName, int CallFrom,char* lpszParent);

void PCS7_Combo_OK_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

void PCS7_Check_OK_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

void PCS7_Binary_OK_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

void PCS7_AnalogPercent_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,double percent);

void PCS7_Analog_OK_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

void PCS7_3Combo_OK_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName);

void PCS7_2Stati_Variable_Changed_V6(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName, double value);

2.1.9.7 Global Script

"Global Script" is not supported for the Web client because there is no separate "Script.exe“ application here.

Global functions must be stored as project functions, they are compiled during publishing.

2.1.9.8 VBS Script

When creating new functions, it is advisable to create them in VBS scripts, since with VBS scripts, there are no differences to be taken into account between WinCC and Web client.

2.1.9.9 Notes

In contrast to WinCC, there are no global, cross-picture tags in C scripts on the Web client.

Cyclic, synchronous functions should be avoided because the runtime on the Web client is greater than in WinCC.

2.1.10 Changing the Language

The templates of the Faceplate Designers are created in three languages (German, English, French), i.e. when the language is changed in WinCC, the texts are displayed in the set language. If you want to add further text elements to a picture and be able to change the language, enter these texts in all relevant languages in the Graphics Designer. The language is changed in the Graphics Designer, using the **View > Language... > Select Language** menu command.

2.1.11 ES Texts for Operator Control of Analog and Binary Values

The comment texts for binary and analog value displays, combo boxes, check boxes and operator logs are fetched from the parameter attributes of the block instances.

Please note that all default texts in ES (s7_shortcut, s7_unit, S7_string_0, S7_string_1) are written in English language.

In previous versions the ES texts were used in the faceplates only for the operator logs, while the texts for the display of faceplates were created in WinCC in three languages.

For migration (e.g. using the IEA), you need to convert these texts in the ES into the language required.

You can use the WinCC Text Library Editor to translate the texts into another language in order to create a multi-language system.
In the SIMATIC Manager, however, you then need to set the same language you have configured for operator and display texts in STEP 7 and in the ES (default is "English") as default for your display devices. This step is prerequisite if you want to prevent the system from overwriting the translated texts the next time you compile the OS.

Important Note:

Considering what was said above (i.e. the default ES texts exist in English language only), you are well advised to create a project library that contains the AS blocks for projects not configured in English language, and there convert the default texts into the language required.

Make sure that the converted text does not exceed the default length. If a longer text passage cannot be avoided, verify its correct display at the faceplate.

For further information on the creation of project libraries, refer to the ES Configuration Manual.

The "s7_unit“ attribute is insignificant for the translation, since the default is here either a space character or a commonly used international short name.

List of Parameter Attributes to be Adapted:

Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
CTRL_PID	AUT_ON_OP		Mode= Manual	Mode= Auto
	DEADB_W	Deadband		
	ER_HYS	ER hysteresis		
	ERH_ALM	ER: HH alarm		
	ERL_ALM	ER: LL alarm		
	GAIN	GAIN		
	HYS	Hysteresis		
	LMNR_IN	OUT		
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_AL		Suppress LL=No	Suppress LL=Yes
	M_SUP_ER		Enable ER Alarm	Suppr. ER Alarm
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	M_SUP_WL		Suppress L=No	Suppress L=Yes
	MAN_HLM	MAN high limit		
	MAN_LLM	MAN low limit		
	MAN_OP	MAN		
	MO_PVHR	Bar HL		
	MO_PVLR	Bar LL		
	OOS		In Service	Out of Service
	OPTI_EN		Disable Optimiz.	Enable Optimiz.
	PV_IN	PV		
	PVH_ALM	PV: HH alarm		
	PVH_WRN	PV: H alarm		
	PVL_ALM	PV: LL alarm		
	PVL_WRN	PV: L alarm		
	SP_HLM	SP high limit		
	SP_LLM	SP low limit		
	SP_OP	SP		
	SP_TRK_ON		No Tracking	Track SP:= PV
	SPBUMPON		SP may bump	SP bumpless
	SPDRLM	Neg. ramp		
	SPEXTSEL_OP		SP= Internal	SP= External
	SPRAMPOF		SP Ramp On	SP Ramp Off
	SPURLM	Pos. ramp		
	TM_LAG	Lag time		
	TN	TI		
	TV	TD		

Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
CTRL_S	AUT_ON_OP		Mode= Manual	Mode= Auto
	BREAK_TM	Break time		
	DEADB_W	Deadband		
	ER_HYS	ER hysteresis		
	ERH_ALM	ER: HH alarm		
	ERL_ALM	ER: LL alarm		
	GAIN	GAIN		
	HYS	Hysteresis		
	LMNDN_OP		Stop	Close
	LMNR_IN	OUT		
	LMNUP_OP		Stop	Open
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_AL		Suppress LL=No	Suppress LL=Yes
	M_SUP_ER		Enable ER Alarm	Suppr. ER Alarm
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	M_SUP_WL		Suppress L=No	Suppress L=Yes
	MAN_HLM	MAN high limit		
	MAN_LLM	MAN low limit		
	MAN_OP	MAN		
	MO_PVHR	Bar HL		
	MO_PVLR	Bar LL		
	MTR_TM	Motor time		
	OOS		In Service	Out of Service
	OPTI_EN		Disable Optimiz.	Enable Optimiz.
	PULSE_TM	Pulse time		
	PV_IN	PV		
	PVH_ALM	PV: HH alarm		
	PVH_WRN	PV: H alarm		
	PVL_ALM	PV: LL alarm		
	PVL_WRN	PV: L alarm		
	RESET		Do Nothing	Reset QMSS_ST
	SP_HLM	SP high limit		
	SP_LLM	SP low limit		
	SP_OP	SP		
	SP_TRK_ON		No Tracking	Track SP:= PV
	SPBUMPON		SP may bump	SP bumpless
	SPDRLM	Neg. ramp		
	SPEXTSEL_OP		SP= Internal	SP= External
	SPRAMPOF		Ascent limit=On	Ascent limit=Off
	SPURLM	Pos. ramp		
	TM_LAG	Lag time		
	TN	TI		
	TV	TD		
DIG_MON	I		Off	On
	SUPPTIME	Suppress time		
	OOS		In Service	Out of Service

Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
DOSE	ACK_TOL_OP		0	Acknowl.
	CANCEL_OP		0	Cancel
	COMP_CHG		Comp. change=Off	Comp. change=On
	DRIB_COR		Dribb. corr.=Off	Dribb. corr.=On
	DRIBB	Dribbling init.		
	DRIBBMAX	Max. dribbling		
	M_SUP_1		Suppr normal =No	Suppr normal=Yes
	M_SUP_2		Suppr over =No	Suppr over =Yes
	M_SUP_3		Suppr under =No	Suppr under =Yes
	MO_PVHR	Bar UL		
	MO_PVLR	Bar LL		
	OOS		In Service	Out of Service
	PAUSE_OP		Process=Continue	Process=Pause
	PDOS_TME	Postdose time		
	POSTDOSE		0	Postdose
	PV_IN	PV		
	RELAXTME	Relax time		
	REVERSE		Reverse=No	Reverse=Yes
	SP_HLM	SP high limit		
	SP_LLM	SP low limit		
	SP_OP	Setpoint		
	SPBUMPON		SP may bump	SP bumpless
	SPEXTSEL_OP		SP= Internal	SP= External
	START_OP		0	Dose=Start
	TOL_N	Lower tol. band		
	TOL_P	Upper tol. band		
ELAP_CNT	HOURS	Hours		
	HOURS_AH	HH alarm		
	HOURS_OP	Preset value		
	HOURS_WH	H alarm		
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	MO_HOUHR	Bar UL		
	MO_HOULR	Bar LL		
	OOS		In Service	Out of Service
	TRACK_OP		0	Preset
FMCS_PID	AUT_ON_OP		Mode= Manual	Mode= Auto
	BREAK_TM	Break time		
	DEADB_W	Deadband		
	GAIN	Gain		
	H_ALM	HH alarm		
	H_WRN	H alarm		
	HYS	Hysteresis		
	L_ALM	LL alarm		
	L_WRN	L alarm		
	LMN	OUT		
	LMN_HLM	LMN high limit		
	LMN_LLM	LMN low limit		
	LMN_OP	MAN		
	LMN_SAFE	LMN safety		
	LMNDN_OP		Stop	Close
	LMNUP_OP		Stop	Open
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_AL		Suppress LL=No	Suppress LL=Yes
	M_SUP_WH		Suppress H=No	Suppress H=Yes

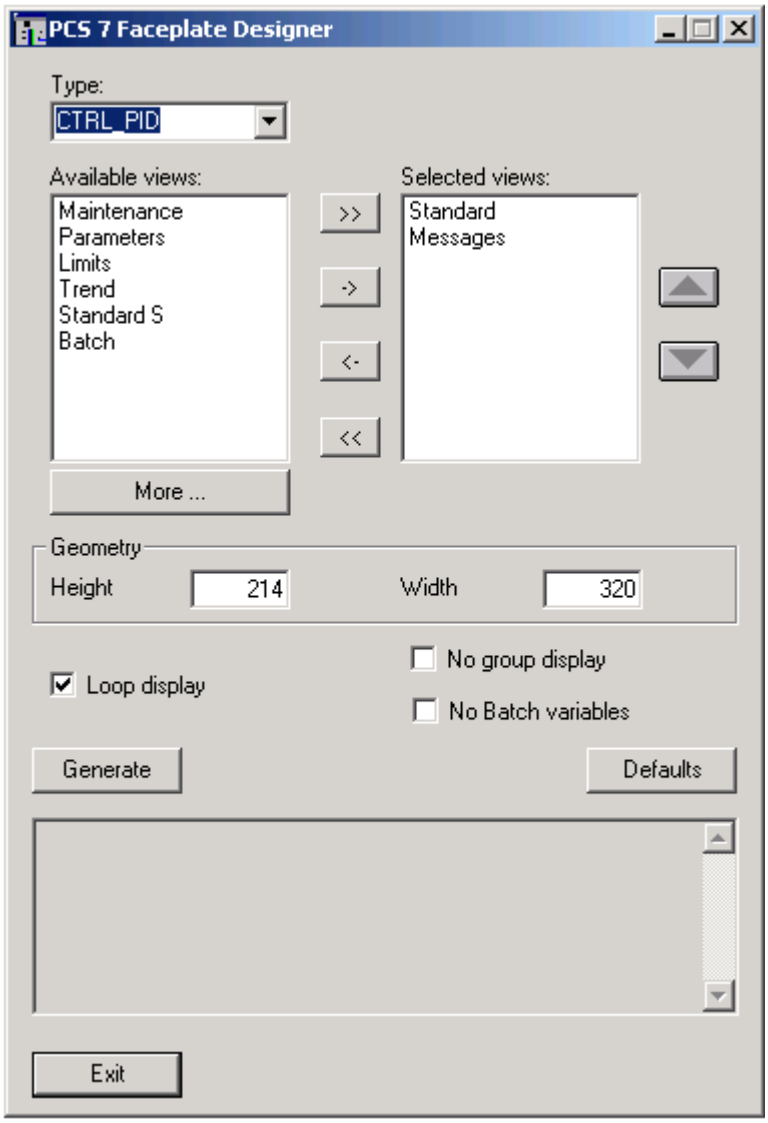
Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
	M_SUP_WL		Suppress L=No	Suppress L=Yes
	MO_PVHR	Bar UL		
	MO_PVLR	Bar LL		
	MTR_TM	MTR time		
	OOS		In Service	Out of Service
	OP_SEL		OP operation=Off	OP operation=On
	OPTI_EN		Optim.=disable	Optim.=enable
	PULSE_TM	Pulse time		
	PV	PV		
	SDB_SEL		SDBParameter=On	SDBParameter=Off
	SP_HLM	SP high limit		
	SP_LLM	SP low limit		
	SP_OP	SP		
	SP_TRK_ON		SP track=Off	SP track=On
	SPBUMPON		SP may bump	SP bumpless
	SPEXTSEL_OP		SP= Internal	SP= External
	TD	TD		
	TI	TI		
	TM_LAG	Time lag		
FMT_PID	AUT_ON_OP		Mode= Manual	Mode= Auto
	BREAK_TM	Break time		
	D_F	deriv. fac.		
	DEADB_W	Deadband		
	GAIN	Gain		
	H_ALM	HH alarm		
	H_WRN	H alarm		
	HYS	Hysteresis		
	L_ALM	LL alarm		
	L_WRN	L alarm		
	LMN	OUT		
	LMN_HLM	LMN high limit		
	LMN_LLM	LMN low limit		
	LMN_OP	MAN		
	LMN_SAFE	LMN safety		
	LMNDN_OP		Stop	Close
	LMNUP_OP		Stop	Open
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_AL		Suppress LL=No	Suppress LL=Yes
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	M_SUP_WL		Suppress L=No	Suppress L=Yes
	MO_PVHR	Bar UL		
	MO_PVLR	Bar LL		
	MTR_TM	MTR time		
	OOS		In Service	Out of Service
	PFAC_SP	Gain		
	PULSE_TM	Pulse time		
	PV	PV		
	SDB_SEL		SDBParameter=On	SDBParameter=Off
	SP_HLM	SP high limit		
	SP_LLM	SP low limit		
	SP_OP	SP		
	SP_TRK_ON		SP track=Off	SP track=On
	SPBUMPON		SP may bump	SP bumpless
	SPEXTSEL_OP		SP= Internal	SP= External
	TD	TD		
	TI	TI		

Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
INTERLOK	I1_1		0	in_1
	I1_2		0	in_2
	I1_3		0	in_3
	I1_4		0	in_4
	I1_5		0	in_5
	I2_1		0	in_6
	I2_2		0	in_7
	I2_3		0	in_8
	I2_4		0	in_9
	I2_5		0	in_10
	OVERWRITE		Overwrite=Off	Overwrite=On
MEAS_MON	HYS	Hysteresis		
	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_AL		Suppress LL=No	Suppress LL=Yes
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	M_SUP_WL		Suppress L=No	Suppress L=Yes
	MO_PVHR	Bar UL		
	MO_PVLR	Bar LL		
	OOS		In Service	Out of Service
	U	PV		
	U_AH	HH alarm		
	U_AL	LL alarm		
	U_WH	H alarm		
	U_WL	L alarm		
MOTOR	AUT_ON_OP		Mode=Manual	Mode=Auto
	MAN_ON		Motor=Stop	Motor=Start
	MONITOR		Monitoring=Off	Monitoring=On
	OOS		In Service	Out of Service
	RESET		0	Error=Reset
	TIME_MON	Mon. time		
MOT_REV	AUT_ON_OP		Mode=Manual	Mode=Auto
	FORW_ON		0	Motor=Forward
	MONITOR		Monitoring=Off	Monitoring=On
	MOT_OFF		0	Motor=Off
	OOS		In Service	Out of Service
	RESET		0	Error=Reset
	REV_ON		0	Motor=Reverse
	TIME_OFF	Mon. time off		
	TIME_ON	Mon. time on		
MOT_SPED	AUT_ON_OP		Mode=Manual	Mode=Auto
	MONITOR		Monitoring=Off	Monitoring=On
	MOT_OFF		0	Motor=Off
	OOS		In Service	Out of Service
	RESET		0	Error=Reset
	SP1_ON		0	Motor=Speed 1
	SP2_ON		0	Motor=Speed 2
	TIME_MON	Mon. time		
OP_A	BTRACK		Bumpless=Off	Bumpless=On
	U	U		
OP_A_LIM	BTRACK		Bumpless=Off	Bumpless=On

Block	Parameter	S7_shortcut	S7_string_0	S7_string_1
	U	U		
OP_A_RJC	BTRACK		Bumpless=Off	Bumpless=On
	U	U		
OP_D	BTRACK		Bumpless=Off	Bumpless=On
	I0		Off	On
OP_D3	BTRACK		Bumpless=Off	Bumpless=On
	I1		0	Switch 1
	I2		0	Switch 2
	I3		0	Switch 3
OP_TRIG	I0		0	Reset
RATIO_P	IN_EX		Internal	External
	MO_U1HR	Bar UL		
	MO_U1LR	Bar LL		
	U1	U1		
	U2	U2		
	U2_HL	High limit U2		
	U2_LL	Low limit U2		
	V_HL	High limit V		
	V_LL	Low limit V		
SWIT_CNT	M_SUP_AH		Suppress HH=No	Suppress HH=Yes
	M_SUP_WH		Suppress H=No	Suppress H=Yes
	MO_VHR	Bar UL		
	MO_VLR	Bar LL		
	OOS		In Service	Out of Service
	TRACK_OP		0	Preset
	V	V		
	VAH	HH alarm		
	VTRACK_OP	Preset value		
	VWH	H alarm		
VALVE	AUT_ON_OP		Mode=Manual	Mode=Auto
	MAN_OC		Valve=Close	Valve=Open
	MONITOR		Monitoring=Off	Monitoring=On
	OOS		In Service	Out of Service
	RESET		0	Error=Reset
	TIME_MON	Mon. time		
VAL_MOT	AUT_ON_OP		Mode=Manual	Mode=Auto
	CLOS_VAL		0	Valve=Close
	MONITOR		Monitoring=Off	Monitoring=On
	OOS		In Service	Out of Service
	OPEN_VAL		0	Valve=Open
	RESET		0	Error=Reset
	SS_POS			
	STOP_VAL		0	Valve=Stop
	TIME_OFF	Mon. time off		
	TIME_ON	Mon. time on		

2.2 Working with the Faceplate Designer

The EXE and DLL files for the Faceplate Designer V6.0 are stored in the "..\WinCC\bin\FaceplateDesigner" directory.



Basic Procedure

1. In WinCC Explorer, call the Faceplate Designer.
2. Enter a name for the new faceplate in the “Type” combo box.

   You can either select a name from the drop-down list of the structured types of the WinCC variable database or assign a new name.
3. In the “Selected views” window on the right, enter the various views of the faceplate you want the Faceplate Designer to generate. You can either select them in the “Available views” window on the left, or click on “Other” to create new view names, however, you may have to configure the multilanguage display of the view list separately. A corresponding note pops up when you generate this new view.

   The procedure is described in the example (Chapter 2.2.1).
4. Configure the picture height and width in the "Geometry" dialog. These, as well as the batch or messages views are preset to default dimensions.

Note when configuring:

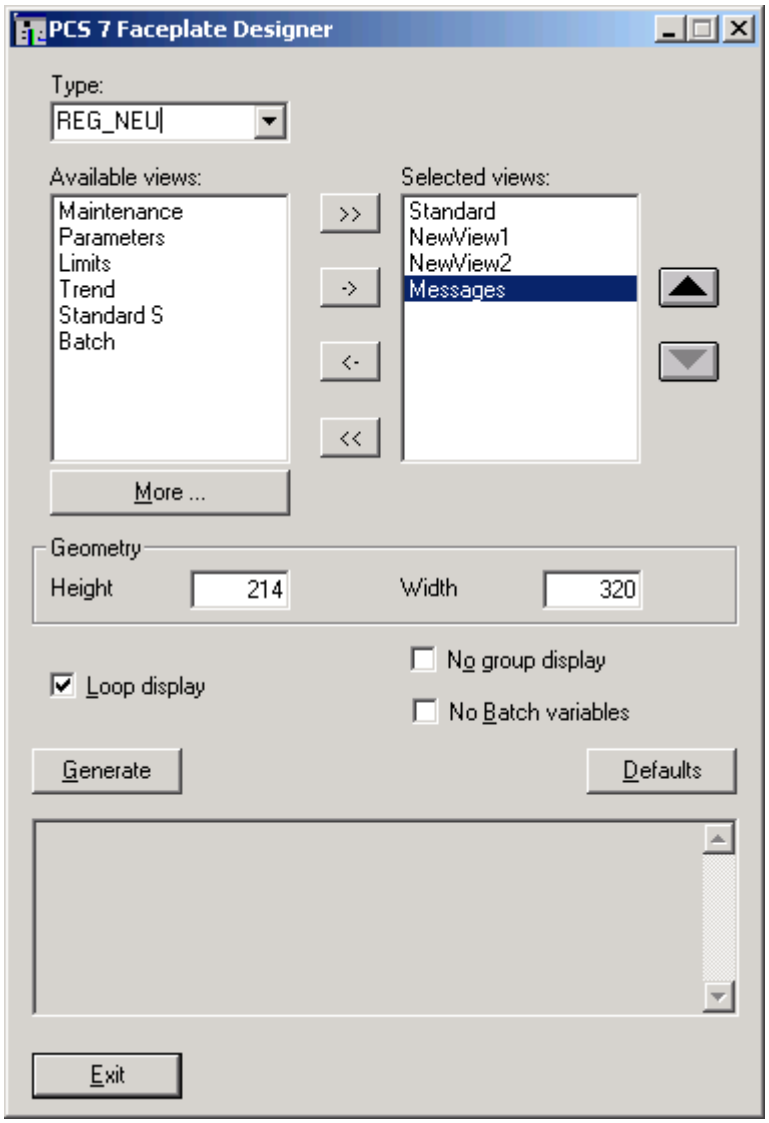
- The picture views are output both in the group display and in the loop display.
- Globally valid pictures are used for Alarm, Batch and Trend.
- You can suppress the generation of a loop view (the button for selecting the loop view is hidden at the same time).
- The default set in the Faceplate Designer is also the dialog language in the WinCC Explorer.
- During its startup, the Faceplate Designer determines the current language of the WinCC user interface, and sets its dialog language accordingly.
- A subsequent change of the WinCC user interface language will be ignored in the open dialog.
- A click on "Default“ restores the standard settings of the Faceplate Designer.
- Picture generation is initiated by clicking on the "Generate" button. The following pictures will be generated (e.g. for a faceplate name “TEST” with a selected view "Selected Views = Standard"):

New picture	From template picture
@PG_TEST.pdl	@PG_%TYPE%.pdl
@PL_TEST.pdl	@PL_%TYPE%.pdl
@PG_TEST_OVERVIEW.pdl	@PG_%Type%_OVERVIEW.pdl
@PG_TEST_VIEWLIST.pdl	@PG_%Type%_VIEWLIST.pdl
@PG_TEST_STANDARD.pdl	@PG_%Type%_%View%.pdl

2.2.1 Example: Creating a New Faceplate for a Controller

2.2.1.1 Creating Templates

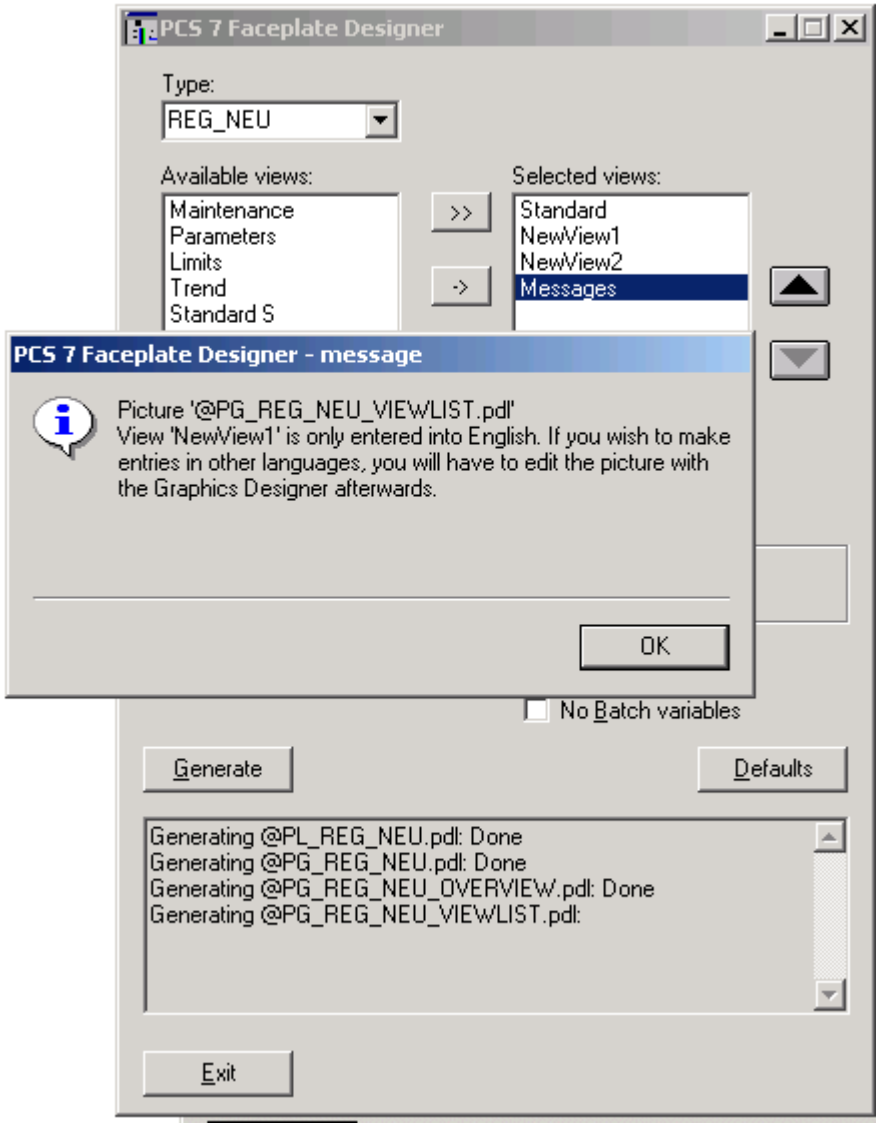
1. In the WinCC Explorer, start the Faceplate Designer.
2. Enter "REG_NEU“ in the "Type" box.
3. Click on "Other..." in "Available views" to create the two new views "NewView1“ and "NewView2“.
4. Click on the "->" (arrow right) button to transfer them to "Selected views".
5. Move the "Message" view to the end of the list by means of the "Cursor down" button.



Click on "Generate" to generate the following picture templates:

New picture	from picture template
@PG_REG_NEU.pdl	@PG_%TYPE%.pdl
@PL_REG_NEU.pdl	@PL_%TYPE%.pdl
@PG_REG_NEU_OVERVIEW.pdl	@PG_%Type%_OVERVIEW.pdl
@PG_REG_NEU_VIEWLIST.pdl	@PG_%Type%_VIEWLIST.pdl
@PG_REG_NEU_STANDARD.pdl	@PG_%Type%_%View%.pdl
@PG_REG_NEU_NEUESICHT1.pdl	@PG_%Type%_%View%.pdl
@PG_REG_NEU_NEUESICHT2.pdl	@PG_%Type%_%View%.pdl

A note informing you of multilanguage aspects pops up during generation.



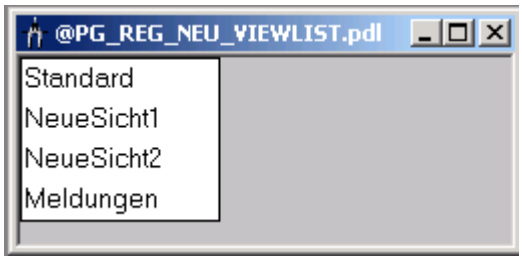
The output box at the bottom lists the generated files.

2.2.1.2 Editing Templates

You can start editing the various views after the picture templates have been generated:

@PG_REG_NEU_VIEWLIST.pdl
@PG_REG_NEU_STANDARD.pdl
@PG_REG_NEU_NEUESICHT1.pdl
@PG_REG_NEU_NEUESICHT2.pdl

The picture "@PG_REG_NEU_VIEWLIST.pdl" requires editing only if the picture block is to be used in a multilingual system.
The sample was generated in German. In the Graphics Designer, when the language is set to "German", you will see the following picture



You will see this picture when you switch to English:

Here you need to adapt the static "Text" properties for the object names "NeueSicht1“ and "NeueSicht2“, e.g. "NewView1“ and "NewView2“.

Do the same in French language.

2.2.1.3 Editing the File @PG_REG_NEU_STANDARD.pdl

We recommend the following procedure to modify the default faceplate of the controller:

1. Open the file "@PG_REG_NEU_STANDARD.pdl"
2. Delete the objects "@Level6" and "@Level5" (see also the description of "Configuring Authorizations ").
3. Open the default faceplate "@PG_CTRL_PID" for the controller and copy/paste it into the picture "@PG_REG_NEU_STANDARD.pdl".

   The "@Level6" and "@Level5" elements for authorizations were included, and are already assigned the direct connections to the operator control elements.

   Note! Special characters in the object names of the copy will be deleted. The "@Level6“ and "@Level5“ objects in the picture "@PG_REG_NEU_STANDARD.pdl" are thus initially named "Level6“ and "Level5“. You must always restore their original name, since the values for these objects are provided by the scripts.
4. Next, you can modify the picture objects and add or delete new objects.

When you delete objects, note that these will receive or pass information via a direct connection. See also: "Documentation of the Standard Faceplates". There you will find the direct connection sequences, including the objects passing information via direct connection. These are as a rule

- "@Level6" and "@Level5", for the transfer of authorizations
- The "Format" object, for the transfer of instance-specific number formats (see also the description of Basic Elements as of Chapter 2.3).

2.2.1.4 Editing the File @PG_REG_NEU_NEUESICHT1.pdl

You can insert and dynamically update objects using the picture template "@PCS7Elements.pdl".

Procedure:

1. In the WinCC Graphics Designer, open the picture "@PG_REG_NEU_NEUESICHT1.pdl" and "@PCS7Elements.pdl".
2. Tile the pictures on the display (menu command: Window > Tile).
3. Copy the picture elements required from the picture "@PCS7Elements.pdl" to the picture "@PG_REG_NEU_NEUESICHT1.pdl".
4. Assign meaningful (object-relevant) object names.
5. Set the positions for the various objects.
6. Interconnect the dynamic attributes of the picture elements to the AS parameters.
7. Create the "authorizations" using the "@Level5" and "@Level6" objects.
8. Save the picture.

2.2.1.5 Dynamic Update of Faceplates

There are various methods of updating the faceplates dynamically:

- The extensions of the required variables are known.

  Call the properties dialog for the object (e.g. a bar graph) to be updated dynamically. In the "Dynamics" column of the properties window, double-click on the bulb icon of the required attribute. You can now enter the extension and the dot, e.g. ".PV_IN“ in the input box

- Variables from the variable list

  The usual procedure used for process pictures takes you to the variable list. However, this list shows all variables. Right-click on the bulb icon in the properties window and select "Variable".

  Mark and then double-click the variable required.
  The dynamic update entry will now, however, show the full variable name. Delete the text up to the extension dot (".“).

2.2.1.6 Creating a Loop View

It is always advisable to create a loop view, even if only one view is required at the faceplate, since the "Picture selection via process tag“ always offer the option of jumping to the loop view.

Procedure:

1. In the Faceplate Designer, generate first the pictures without a loop view, in order to generate the group picture view without the loop view selection button.
2. The repeat the generation with loop picture view. Answer the prompt for all pictures with "No", except for the prompt "Overwrite Loop Picture" prompt.

2.2.1.7 Generating an Additional View

To generate a new view after you have modified the existing views, we suggest you take the following steps:

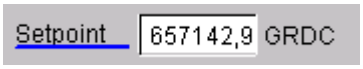
1. In the Faceplate Designer, enter the type names and selected views exactly in the same way as you did during the initial generation.
2. Enter the new view and start generation.

   During the generation, you are prompted to overwrite existing files.
3. Answer these prompts with "No".

2.3 Basic Elements

All basic elements are stored in the picture model "@PCS7Elements.pdl" which is stored in the "..\Wincc\options\pdl\FaceplateDesigner_V6" directory and copied to the project by the OS Project Editor.

2.3.1 Analog Value Display and Operator Control of Analog Values



Object type :	PCS7_AnalogValue
Object name:	PCS7_AnalogValue1, PCS7_AnalogValue2, PCS7_AnalogValue3
Picture name:	@PCS7Elements.pdl

The two objects “PCS7_AnalogValue1“ and "PCS7_AnalogValue2“ are identical and are here configured as an example.

- “PCS7_AnalogValue1“ for operator control of the analog value and
- “PCS7_AnalogValue2“ for the analog value display.

The floating-point format of the I/O field is used for both objects. You can use these objects for the display or input of numbers which are not changed by the process, since the floating-point does not cause any disturbance in this case.

However, it is advisable to use the "PCS7_AnalogValue3“ object, which was created with the new AdvancedAnalogDisplay. In this case, only the numbers float, whereas the decimals can be configured instance-specific.

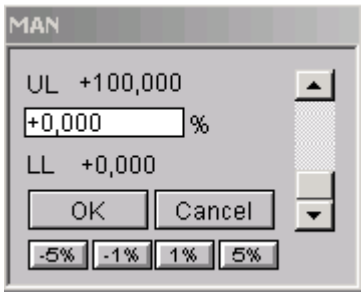
When used solely as analog value display, set the "Operator control enable“ attribute to "FALSE“ and the "Background color“ property to "gray".

When used for operator control of analog values, a mouse-click calls the script "PCS7_OpenInputBoxAnalog_V6“.

The script transfer parameter CallFrom = 0

"PCS7_OpenInputBoxAnalog_V6(lpszPictureName,lpszObjectName,0);“

calls the operator control picture "@PCS7_BedAnalog.pdl“. This operator control picture contains a percentage adjustment slide, which is used for operator control of analog values with control limits.



The script transfer parameter "CallFrom = 1"

"PCS7_OpenInputBoxAnalog_V6(lpszPictureName,lpszObjectName,1);“

calls the operator control picture "@PCS7_BedAnalog_NL.pdl“. This operator control picture does not contain a slide and is used for operator control of analog values without control limits.

To transfer changes of process-controlled operator control enabling to the OS picture directly at the time of change, you can also call

- the "PCS7_OpenInputBoxAnalog_V6(lpszPictureName,lpszObjectName,10);“ script for the operator picture with limits

  and

- the "PCS7_OpenInputBoxAnalog_V6(lpszPictureName,lpszObjectName,11);“ script for the operator picture without limits

in the "Operator control enabling" property of the "PCS7_AnalogValue“ object.

This covers situations in which operator control is disabled precisely for this value in the operator picture selected exactly at this moment, so that the operator can no longer control the value.

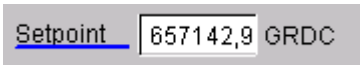
Properties	Function
ProcessValue/OutputValue	Structured element of the operator controlled analog value.
ProcessValue/VisibleValue	Structured element of the analog value that is dynamically updated.
Limits/UpperLimit	Structured element of the “High” limit of the operator controlled analog value
Limits/LowerLimit	Structured element of the “Low” limit of the operator controlled analog value
Other/operator control enable	Operator control enable on the AS side, e.g. set to "Q_SP_OP" or permanently to "No", if only one for one display.
Other/Display_Analogvalue	Must always be set visible.
Other/Password	Should not be used, since this is handled by the authorization properties.
Other/Display_Line	Display of the blue line.
Other/Line thickness	The line thickness can be customized.
Other/ Open_BedBox_from_Bar	Properties for indirect calls of the operator picture "@PCS7_BedAnalog.pdl“, e.g. via bars.
Font/Text	Labeling of the I/O field (setpoint).
Font/ Unit	Unit of the analog value, dynamically via .MEMBER#unit, e.g. .SP_OP#unit
Font/X-Alignment_Text	Aligned left, if several I/O fields are arranged one below the other (due to the alignment). Right alignment is advisable for single analog values.
Colors/Fontcolor	Selection of the font color.
Colors /Linecolor	Selection of the line color.
Colors /Background_Value	Selection of the background color "Value", updated dynamically by the script upon the event "Changes of enable operator control“.
Colors /Font color_Value	Selection of the font color "Value".
Geometry/Width_Analogvalue	Overall width of the analog field.
Geometry/Width_Line_Text	Line length, text can be customized.
Geometry/PositionX1_Line_Text	Must be adapted when the text and line length is changed.
Geometry/PositionY1_Line	Line height compared to the font.

The "Font/Text“ property is interconnected by default with the sample parameter ".SP_OP#shortcut“, meaning that the display text of the block icon as well as the title of the operator picture for analog control is fetched from the AS (s7_shortcut). Variables without display text (e.g. PV_IN) must be deleted from the interconnection and configured directly in WinCC (in three languages, if required).

The script "PCS7_OpenInputBoxAnalog_V6" is also called when the "Open_BedBox_from_Bar“ property is changed. In this case, operator control enable are also verified first.

The script can also be used for indirect operator control. The corresponding operator picture is called, for example upon the event "Click on the setpoint bar graph“. For more information, refer to the description of the double bar graph that displays both the setpoint and the process value. A change of the property is there diverted to the "Open_BedBox_from_Bar“ property by means of direct connection.

2.3.2 "AdvancedAnalogDisplay"



Object type: PCS7_AnalogValue

Object name: PCS7_AnalogValue3

Picture name: @PCS7Elements.pdl

The number format of the AdvancedAnalogDisplay is available for situations in which a floating-point format is not desirable (in particular for the decimals). Here, the number format can be defined for specific instances via the block icon
See section 2.1.6 "Configuring Number Formats ".

The description of the objects in the user object and the property function is the same as in Chapter 2.3.1.
An additional "Format" property is available. The number format is controlled by means of direct connection, if instance-specific number formats are required.

2.3.3 StaticText

The static text is used to label check boxes, I/O buttons etc.

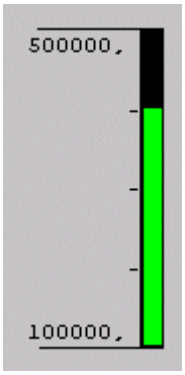
Object type: PCS7_StaticText

Object name: StaticText

Picture name: @PCS7Elements.pdl

Character set Arial, font size 12

2.3.4 Simple Analog Bar Graph



Object type: PCS7_BarStandard_1

Object name: BarStandard_1

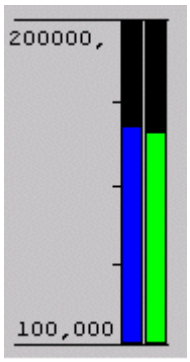
Picture name: @PCS7Elements.pdl

Properties	Function
Links/PV	Structured element of the analog value, dynamically updated as bar graph.
Links/Range_LL	Structured element for "Bar graph low limit" (initial measurement value).
Links/Range_UL	Structured element for "Bar graph high limit" (final measurement value).
Other/PvunderLimit	Underflow measuring range. The user may not change the process variable.
Other/PvoverLimit	Overflow measurement range. The user may not change the process variable.
Colors/Bar graph color_PV	Color of the bar graph.

The scripts (call of the script "PCS7_UpdateBar_V6.fct") available at each of the interconnected PV, RANGE_LL, RANGE_UL provide an arrow indication at the bar limits when a change in the values causes the measurement to exceed range.

2.3.5 Double Analog Bar Graph

The double bar graph displays the setpoint and the process variable at the same time.



Object type: PCS7_BarStandard_2

Object name: BarStandard_2

Picture name: @PCS7Elements.pdl

Properties	Function
Links/PV	Structured element of the process variable, dynamically updated in the bar graph.
Links/SP	Structured element of the setpoint, dynamically updated in the bar graph
Links/ Range_LL	Structured element for "Bar graph low limit" (initial measurement value).
Links/ Range_UL	Structured element for "Bar graph high limit" (final measurement value).
Other/ SPunderLimit	Display of measurement range underflow. The user may not change the setpoint.
Other/PVunderLimit	Display of measurement range underflow. The user may not change the process variable.
Other/SpoverLimit	Display of measurement range overflow. The user may not change the setpoint.
Other/PVoverLimit	Display of measurement range overflow. The user may not change the process variable.
Colors/ Barcolor_SP	Bar graph color for the setpoint.
Colors/Barcolor_PV	Bar graph color for the process variable.

When the operator control enable property of the bar graph display is set "TRUE“, this setting is set briefly to "FALSE“ and then immediately reset to "TRUE“ by the script when the operator clicks the bar graph.

The "operator control enable" property can then be passed to the corresponding basic element "PCS7_AnalogValue" (property "Open_BedBox_from_Bar“) by means of direct connection, were it initiates the call of the analog display for operator control of the setpoint (see also the description of operator control of analog values, Chapter 2.3.1).

The bar graph display is supplemented by arrow indicators for values above and below the bar graph limits.

The scripts (call of the script "PCS7_UpdateBar.fct") stored in each one of the interconnected properties PV, SP, RANGE_LL and RANGE_UL displays arrow indicators at the bar graph limits to indicate measuring range errors when values change.

2.3.6 Horizontal Bar Graph

The horizontal bar graph can be used to display manipulated variables, for example.

Object type: PCS7_BarStandard_3

Object name: BarStandard_3

Picture name: @PCS7Elements.pdl

The "Operator control enable" property can then be passed to the corresponding basic element "PCS7_AnalogValue" (property "Open_BedBox_from_Bar“) by means of direct connection, were it initiates the call of the analog display for operator control of the setpoint (see also the description of operator control of analog values, Chapter 2.3.1).

The bar graph display is supplemented by arrow that indicate values above and below bar graph limits.

Properties	Function
Other/Barcolor	Color of the bar graph
Other/PvunderLimit	Display of measurement range underflow. The user may not modify the process value.
Other/PvoverLimit	Display of measurement range overflow. The user may not modify the process value.
Other/Unit	Unit of the manipulated variable.
Links/RANGE_LL	Start of the measurement range of the bar graph display.
Links/50PWERT	50%-display is calculated by a script when RANGE_LL or RANGE_UL changes.
Links/RANGE_UL	End of the measurement range of the bar graph display.
Links/PV	Structured element of the manipulated variable, dynamically updated as bar graph.

2.3.7 Bar Graph "Limit Value Display"

Object type: PCS7_BarLimits
Object name: BarLimits
Picture name: @PCS7Elements.pdl

Properties	Bar graph property	Function
Limits/RANGE_UL	Maximum value and process connection	End of the bar graph measurement range (.MO_PVHR
Limits/RANGE_LL	Minimum value, zero value and high limit RH5	Start of the bar graph measurement range (.MO_PVLR)
Limits/VALUE_AH	High limit AH	Limits display "Alarm high" (.PVH_ALM)
Limits/VALUE_WH	High limit WH	Limits display "Warning high" (.PVH_WRN)
Limits/VALUE_WL	Low limit WL	Limits display "Warning low" (.PVL_WRN)
Limits/VALUE_AL	Low limit AL	Limits display "Alarm low" (.PVL_ALM)
Limits/@HighLimitTH	High limit TH	Control of the low limit of the second to last segment. This property is only visible in the setup dialog.
Limits/@LighLimitRH4	High limit RH4	Control of the low limit of the last segment. This property is only visible in the setup dialog.
Limits/@ Bar graph colorRH4	Bar graph color RH4	Color setting of the second to last segment. This property is only visible in the setup dialog.
Limits/@ Bar graph color RH5	Bar graph color RH5	Color setting of the last segment. This property is only visible in the setup dialog.
Colors/ColorAlarm	Bar graph color AH	
Colors/ColorWarning	Bar graph color WH	
Colors/ ColorBackground	Bar graph color TH	Color of the middle segment.

The script "PCS7_UpdateBarLimits_V6" is called when the low limit values "VALUE_WL", "VALUE_AL" and "RANGE_LL" change, since the "Alarm" and "Low Limit Warning " bar graph display cannot be generated by means of the standard bar graph. A script is not required for the high limit values.

2.3.8 "Message Suppression Display"

Object type: PCS7_MSG_LOCK

Object name: MSG_LOCK

Picture name: @PCS7Elements.pdl

Properties	Function
User-defined2/ actual status	Control of "Status display2". Messages with a diagonal strike-out lock requests from the AS (.MSG_LOCK) 
User-defined2/ actual status1	Control of "Status display3". Crossed-out messages are locked. Feedback from WinCC (.QMSG_SUP) 
User-defined2/ Display1	"Status display3" is shown and overlays "Status display2" (.QMSG_LOCK)

2.3.9 "Batch Occupied" Display

Object type: PCS7_OCCUPIED

Object name: OCCUPIED

Picture name: @PCS7Elements.pdl

Properties	Function
User-defined2/ actual status	Control of status display 1 / Indicates that a block is occupied by a batch. Structured element (.OCCUPIED).

2.3.10 Acknowledgment of Messages from the Selected Block

Object type: Button

Object name: ButtonQS

Picture name: @PCS7Elements.pdl

This group message acknowledgment button is the same as the one of the standard button set "@Buttons11.pdl", but has a different script for instance-specific message acknowledgment and works only in combination with a group display.

2.3.11 "Locked " Display Block (Valve, Motor)


Object type: Status display

Object name: Lock

Picture name: @PCS7Elements.pdl

Properties	Function
Status/actual status	Indicates whether a block is locked. Structured element (.LOCK)

2.3.12 Group Display

Object type: Group display

Object name: EventState

Picture name: @PCS7Elements.pdl

Properties	Function
Other/CollectValue	Display of alarm and warning states. Structured element (.EventState)

2.3.13 Operator Control of Binary Values with Check Box_R

@PCS7_BedCheck.pdl

Object type: CHECKBOX_R

Object name: Checkbox_R1

Picture name: @PCS7Elements.pdl

The picture "@PCS7_BedCheck.pdl" is called by the "PCS7_OpenCheckbox_V6" script.

Properties	Function
Other/ DisplayActivWith	DisplayActivWith = 1 → The value 'TRUE' is written to the binary variable when the check box is set. DisplayActivWith = 0 → The value 'FALSE' is written to the binary variable when the check box is set.
Other/Enable Operator Control	Operator control of the check box enabled.
Interconnection/Input	Title of the operator control picture for the "@PCS7_BedCheck.pdl" check box.
Interconnection/Variable	Interconnection to the binary structured element, e.g. (.MSG_LOCK).
Interconnection/NegatedVariable	Check box that is clicked; may not be modified by the user.
Interconnection/Read_Text_From_AS	Text is fetched from "String_0" or "String_1" and assigned to the "Input" property.
Interconnection/CaptionCheckBoxOn	YES = The text of "Input" is displayed on the right side of the check box. NO = The check box text is not displayed if insufficient space is available.

The internal option "Inverted Variable“ is selected by clicking on the check box. The status of the binary variable to be controlled by the operator is determined and written back to "Inverted Variable" ("Interconnection/NegatedVariable" property). When the "Interconnection/Variable“ property changes and when a picture is being selected, a script is executed to update the display of the "NegatedVariable" (see also the Check_Box flow chart). The "Interconnection/Variable“ property must be set by the configuration to "unequal 1“ and "unequal 0“ (0x8), to ensure that the script is always executed when a picture is selected.

The text for labeling the check boxes is fetched from the "String_0" and "String_1" parameter attributes of the block instance.

Notes on labeling the check box:

- If the property "Read_Text_From_AS" is set to 1, the full text of "String_1" is displayed on the right side of the check box when "DisplayActivWith = 1".
- When "DisplayActivWith = 0", the full text of "String_0" is displayed on the right side of the check box.
- The text configured at the “Input“ object is displayed if the "Read_Text_From_AS" property is set to 0. This must be set in particular for the display of output parameters, since "String_0" or "String_1" do not exist in this situation and a script error would otherwise be generated. Operator control of the check box may not be enabled In this situation.

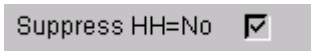
Notes on labeling single-selection buttons:

- When the text of the strings ("String_0" and "String_1") contains a "=“ character, the string text after the "=“ forms the label of the single-selection buttons.
- If the text of the strings does not contain a "=“ character, the full text forms the label of the single-selection buttons.

Notes on the title of operator control pictures:

- If the text of the strings ("String_0" and "String_1") contains a "=“ character, the text to the left of the "=“ forms the title of the process control picture, i.e. "DisplayActivWith = 1" > "String_1" text and "DisplayActivWith = 0" > "String_0" text.
- If the text of the strings does not contain a "=“ character, the check box labeling text is used.

2.3.14 Operator Control of Binary Values with "Check Box_L"

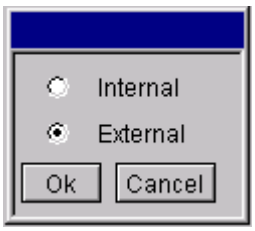


The functionality is similar to that of "CheckBox_R", the difference being that the text is displayed left aligned.

Even if the text is not displayed, both variants (right or left aligned text) are still required, depending on whether there is a further operator control element positioned on the left or right side of the check box (e.g. for alarm limits of MEAS_MON).

For information on properties and functions refer to Chapter 2.3.13.

2.3.15 Operator Control of Binary Values with Combo Box



@PCS7_BedCombo.pdl

Object type: PCS7_COMBOBOX

Object name: COMBOBOX1

Picture name: @PCS7Elements.pdl

The picture "@PCS7_BedCombo.pdl" is called by the script "PCS7_OpenComboBox_V6.fct".

The texts are fetched from "String_0" and "String_1".

Properties	Function
Colors/BackColor_Text1	Background color Text1
Colors/BackColor_Text2	Background color Text2
Font/Text1	Display text for Text1 (fetched from the AS).
Font/Text2	Display text for Text2 (fetched from the AS).
Displays/Display_Text1	Display text1 (controls which value is selected in the display).
Displays/Display_Text2	Display text2 (controls which value is selected in the display).
Other/ OP_enabled_Text1	Operator control enable for Text1.
Other/ OP_enabled_Text2	Operator control enable for Text2.
Other/ operator control enable	Operator control enable for entire combo box for control via "@Level5/6".
Other/ Password_Text1	Authorization for Text1 (not used).
Other/ Password_Text2	Authorization for Text2 (not used).
Links/Write_Variable1	Structured element to which selection Text1 is written.
Links/Write_Variable2	Structured element, to which selection Text2 is written.
Links/Display_Variable1	Structured element; displays the first binary state (Text1).
Links/Display_Variable2	Structured element; displays the second binary state (Text2).
Parameters/ Write_value_Text1	Specifies which value is written to the structured element "Write_Variable1“ with selection Text1 ('TRUE' or 'FALSE').
Parameters/Display_Text1_with	Specifies which value ('TRUE' or 'FALSE) the structured element "Display_Variable1" uses to display Text1.
Parameters/ Write_value_Text2	Specifies which value is written to the structured element "Write_Variable2“ with selection Text2 ('TRUE' or 'FALSE').
Parameters/Display_Text2_with	Specifies which value ('TRUE' or 'FALSE) "Display_Variable2" uses to display Text2.

The texts for labeling the check box is fetched from the parameter attributes "String_0" and "String_1" of the block instance.

If the text of the strings contains a "=“ character, the text after the "=“ is used to label the check box and the text on the left side of the "=“ forms the title of the process control picture.

If the text of the strings does not contain a "=“ character, the full text is used to label the check box. In this case, the title of the process control picture is filled with space characters.

Notes on Operator control enable:

The background of the combo box text can be grayed out in order to indicate or highlight that operator control is not enabled. This is done by transferring the "@Level5" or "@Level6" background color to "BackColor_Text1" in this object by means of one direct connection, and the "BackColor_Text1" background color to "BackColor_Text2" by means of a further direct connection. Both background colors should then be gray by default.

However, this option is only available if the various switching states of the combo box are displayed by a white background, so that the authorization status indication toggles between gray and white.

Example of "Manual/Auto" changeover of the controller mode:

The combo box uses the background color to identify the set operating mode "Manual/Auto" (White = Manual, Green = Auto)

Because of this, the background color of the display cannot be toggled to gray in order to indicate disabled operator control. The dropdown button of the combo box is therefore hidden in addition to indicate this status.

Operator control enable "Yes" 

Operator control enable "No" : 

2.3.16 Operator Control of Binary Values with Combo Box (3ComboBox)

@PCS7_3BedCombo.pdl

Object type: PCS7_3COMBOBOX

Object name: 3COMBOBOX1

Picture name: @PCS7Elements.pdl

The picture "@PCS7_3BedCombo.pdl" is called by the "PCS7_Open3ComboBox_V6.fct" script.

The texts are fetched from "String_0" and "String_1".

Properties	Function
Colors/BackColor_Text1	Background color Text1.
Colors/BackColor_Text2	Background color Text2.
Colors/BackColor_Text3	Background color Text3.
Font/Text1	Display text for Text1 (fetched from the AS).
Font/Text2	Display text for Text2 (fetched from the AS).
Font/Text3	Display text for Text3 (fetched from the AS).
Displays/Display_Text1	Display Text1 (controls which value is selected for the display).
Displays/Display_Text2	Display Text2 (controls which value is selected for the display).
Displays/Display_Text3	Display Text3 (controls which value is selected for the display).
Other/ OP_enabled_Text1	Operator control enable for Text1.
Other/ OP_enabled_Text2	Operator control enable for Text2.
Other/OP_enabled_Text3	Operator control enable for Text3.
Other/Op_Enable	Enables operator control of the complete combo box for control via "@Level5/6".
Other/Password_Text1	Authorization for Text1 (not used).
Other/Password_Text2	Authorization for Text2 (not used).
Other/Password_Text3	Authorization for Text3 (not used).
Links/Write_Variable1	Structured element to which selection Text1 is written.
Links/Write_Variable2	Structured element to which selection Text2 is written.
Links/Write_Variable3	Structured element to which selection Text2 is written.
Links/Display_Variable1	Structured element displaying the first binary state (Text1).

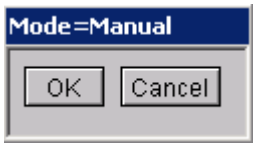
Properties	Function
Links/Display_Variable2	Structured element displaying the second binary state (Text2).
Links/Display_Variable3	Structured element displaying the second binary state (Text3).
Parameters/ Write_value_Text1	Specifies which value is written to the structured element "Write_Variable1“ with selection Text1 ('TRUE' or 'FALSE').
Parameters/Display_Text1_with	Specifies which value ('TRUE' or 'FALSE') "Display_Variable" uses to display Text1
Parameters/ Write_value_Text2	Specifies which value is written to the structured element "Write_Variable2“ with selection Text2 ('TRUE' or 'FALSE').
Parameters/Display_Text2_with	Specifies which value ('TRUE' or 'FALSE') "Display_Variable" uses to display Text2
Parameters/ Write_value_Text3	Specifies which value is written to the structured element "Write_Variable3“ with selection Text3 ('TRUE' or 'FALSE').
Parameters/Display_Text3_with	Specifies which value ('TRUE' or 'FALSE') "Display_Variable" uses to display Text3

The texts used to label the check box are fetched from the parameter attributes "String_0" and "String_1" of the block instance.

If the text of the strings contains a "=“ character, the text after the "=“ forms the label of the check box and the text on the left side of the "=“ forms the title of the process control picture.

If the text of the strings does not contain a "=“ character, the full text forms the label of the check box. In this case, the title of the process control picture is filled with space characters.

2.3.17 Button Control of Binary Values and Color Change



@PCS7_BedBinaer.pdl

Object type:	PCS7_BinOp
Object name:	BinOp0 / BinOp1
Picture name:	@PCS7Elements.pdl

Single or double button operation of binary values can be set at the "CMD2Steps" parameter.

CMD2Steps = Yes
The "PCS7_OpenInputBoxBin_V6.fct" script calls the picture "@PCS7_BedBinaer.pdl" when the operator selects "Auto" or "Manual" mode. By clicking on "OK" in this picture, the value determined via "Write_Value" is written to the variable linked to "Write_Variable" (see also the "BinOp.vsd“ flow chart).

CMD2Steps = No
The value determined in "Write_Value" is written directly to the variable linked to "Write_Variable" when the operator clicks STOP or RUN.

The labeling text for the buttons is fetched from the parameter attributes "String_0" and "String_1" of the block instance.

If the text of the strings contains a "=“ character, the text after the "=“ is forms the label of the buttons, whereas the full text is used to form the title of the operator control picture.

If the text of the strings does not contain a "=“ character, the full text forms the label of the buttons.

Properties	Function
Colors/Button_Colour	Background color for text, if active and not operator controlled.
Other/Op_Enable	Enables operator control
Other/DisplayActive	Properties of the "Button On" display. May not be modified by the user; controlled by script.
Links/Write_Variable	Structured element to which the selection text is written.
Links/Display_Variable	Structured element indicating the first binary state (Text).
Parameters/ Write_value	Defines which value is written to the structured element "Write_Variable“ with selection Text ('TRUE' or 'FALSE').
Parameters/ Display_is_active_with	Defines the value ('TRUE' or 'FALSE') which "Display_Variable" uses to indicate the text in the selected color (display active and operator control disabled).
Parameters/ButtonText	The display text may not be modified; controlled by script.
Parameters/CMD2Steps	1- or 2-button control

2.3.18 Operator Control of Binary Values with Button

@PCS7_BedBinär.pdl

Object type:	PCS7_ButtonBit
Object name:	ButtonBit
Picture name:	@PCS7Elements.pdl

Same functions as described in Chapter 2.3.17, except:

- No color change
- The button always indicates operator control mode (it has no indication of "Button On“).

Single- or two-button operation of binary values can be set at the "CMD2Steps" parameter.

- CMD2Steps = Yes
  When the operator selects "Reset", the "PCS7_openinputboxbin_V6.fct" script calls the picture "@PCS7_BedBinär.pdl". By clicking on "OK" in this picture, the value determined in "Write_Value" is written to the variable linked to "Write_Variable".
- CMD2Steps = No
  The value determined in "Write_Value" is written directly to the variable linked to "Write_Variable" when "Reset" is clicked.

2.3.19 Status Display with two Alternatives

Object type: PCS7_Status_2_Alternative

Object name: Status_2_Alternative

Picture name: @PCS7Elements.pdl

Properties	Function
Links/Link	Interconnection with the binary structured element, e.g. (.MSG_LOCK)
Others/ Read_Text_From_AS	Text fetched from the AS for labeling the status display ("String_0" and "String_1").
Others/Display	May not be changed by the user; must remain 'TRUE'.
Others/Op_Enable	Enables operator control
Font/Text_On	Description of the 'True' status.
Font/Text_Off	Description of the 'FALSE' status.
Colors/BackColor_OFF	Background color for the 'Off' status.
Colors/BackColor_ON	Background color for the 'On' status.
Colors/TextColor_OFF	Text color for the 'Off' status.
Colors/TextColor_ON	Text color for the 'On' status.
Displays/Off	Display of the static text "T_off". May not be modified by the user; controlled by script.
Displays/On	Display of the static text "T_on". May not be modified by the user; controlled by script.

The "PCS7_2Stati_Variable_Changed_V6.fct" script is called when the variable interconnected to the "Links/Link" property changes.

This script controls the display of "T_off" and "T_on", based on the status of "Links/Link".

The status texts can be fetched from the AS (Read_Text_From_AS = TRUE) by setting the link to an input parameter (e.g. AUT_ON_OP). "String_0" or "String_1" are usually not available for the output parameters, i.e. you must write the text directly to the object (Text_On, Text_Off) (Read_Text_From_AS = FALSE).

2.3.20 Status Display with n Alternatives

Object type: PCS7_Status_1_v_n

Object name: Status_1_v_n

Picture name: @PCS7Elements.pdl

Properties	Function
Links/ Link	Interconnection with the binary structured element (.MSG_LOCK)
Others/ Read_Text_From_AS	"Text_On" label for the status display, fetched from the AS (String_1).
Others/Display	May not be modified by the user; controlled by script
Others/Op_Enable	Enables operator control
Font/Text_On	Description of the 'True' status.
Colors/BackColor_ON	Background color for the 'On' status.
Colors/TextColor_ON	Text color 'On' status
Masks/ Mask	Value used to control visibility of the text box.

For status displays with n alternatives, the number of alternatives required determines the number of objects of the type **Status_1_v_n** to be arranged in an overlay stack.

The "PCS7_1vnStati_Variable_Changed_V6.fct" script is called when the variable which is interconnected with the "Links/Link" property changes.

In the "Masks/Mask" property, you can define the value to be used to display the relevant object. This option is suitable for alternative control by means of INTEGER or REAL values, for example.

Set the value in "Mask" to 1 for all objects of an alternative control of multiple binary values. You can set the priority for all active binary values in the "Level“ property. WinCC knows the levels 0 to 15, sorted in ascending priority.
Example: Of two objects, one is assigned to level 4 and the other to level 5. Both of them are set visible and arranged in an overlay stack. The level 5 object is thus displayed, while the level 4 object remains hidden.

When you create a link to an input parameter (e.g. REV_ON), the status text can be fetched from the AS (Read_Text_From_AS = TRUE). A "String1" usually does not exist for output parameters and you therefore need to write the text directly to the object (Text_On) (Read_Text_From_AS = FALSE).

2.3.21 Status Display "Valve"

The following valve status display was taken from PCS 7 V5.

As of PCS 7 V6.x, it is advisable to use the "Extended Status Display“ function to create the status displays. Refer to the WinCC process control options.

Object type: PCS7_Valve_Stat

Object name: Valve_Stat1,Valve_Stat0,Valve_Stat2

Picture name: @PCS7Elements.pdl

Properties	Function
Other/Operator enable	Operator control enable
Other/Display	Display of the user object.
Link/VariableLink	Interconnection to the structured element, if the status control is to be handled by means of REAL or INTEGER values.
State/Index	Manual setting of the status display for binary controls via "State/Display".
State/Display	Status control by means of binary variables.

The user object "Valve_Stat" can be used in two ways.

- The first option is to use the object for a standard status display and to control the states via the "Link/VariableLink“ property. "State/Display" must have been set to "Yes“. These three states described above are only possible, however, if the valve states are available in a single variable; this is usually not the case.
- The second option is to create a "Valve_Stat" object for each one of the three states and to arrange them in an overlay stack. The status to be displayed is set in "State/Index". The link to the corresponding structured element is set in "State/Display". This variant is used for the VALVE faceplate.

2.3.22 Status Display "Motor"

Object type: StatusDisplay

Object name: Motor

Picture name: @PCS7Elements.pdl

Standard status display, with two alternatives and control via the "Status/ActualStatus" property.

2.3.23 Permission

Object type: Permission

Object name: Permission_Setpoint

Picture name: @PCS7Elements.pdl

The "Permission" object is designed for creating global permissions for an operator controlled "Setpoint" element.

There are operator controlled elements which are subject to different procedures for the verification of permissions.

Example:

- The permission for operator control of the "Setpoint" is subject to verification by the user administration in WinCC, which is always performed when new users log in (controlled via "@Level5" or "@Level6").
- The permission for operator control of the "Setpoint" is furthermore subject to verification by means of AS variables (Q_SP_OP, permission granted only for internal adjustment of the setpoint). This verification is also performed for write access to these variables.

At the FMCS_PID, the QPARF parameter of the AS is also called for the verification of permissions.

Configuration:

1. Creating the direct connection from "@Level5“ or “@Level6“, "Operator control enable“ property to "Permission“ object, "Level_Source“ property.
2. C-script for changing the "Level_Source“ property:

```
BOOL bTag1 =!GetTagBitWait(".QPARF");
BOOL bTag2 =GetTagBitWait(".Q_SP_OP");
if (bTag1 &&  bTag2 && value)
{
SetPropBOOL(lpszPictureName,lpszObjectName,"Target_Permission",TRUE);
SetPropWord(lpszPictureName,lpszObjectName,"Target_BackgroundColor",CO_WHITE);
}
else
{
SetPropBOOL(lpszPictureName,lpszObjectName,"Target_Permission",FALSE);
SetPropWord(lpszPictureName,lpszObjectName,"Target_ BackgroundColor ",CO_LTGRAY);
}
SetPropBOOL(lpszPictureName,lpszObjectName,"Level_Target",value);
```

   This script is called when a new user logs in or when a picture is selected.

   The AS variables, however, must always be included in the verification of authorizations. In this case: "QPARF" and "Q_SP_OP".

   If all three criteria are met, the "TARGET_PERMISSION“ property is set 'TRUE', the "TARGET_BackgoundColor“ property is set "white".

   The value of the "Level_Source“ property is transferred to the "Level_Target“ property.

   "Level_Target“ is defined for the interconnection of the direct connections to further objects.

3. C script under the "Tags" property

   This script is called by the variables included in the verification of authorizations (QPARF und Q_SP_OP).

```
BOOL bTag1 =!GetTagBitWait(".QPARF");
BOOL bTag2 =GetTagBitWait(".Q_SP_OP");
**  BOOL bLevel = GetPropBOOL(lpszPictureName,"@Level5","Operation");
if (bTag1 &&  bTag2 && bLevel)
{
SetPropBOOL(lpszPictureName,lpszObjectName,"Target_Permission",TRUE);
SetPropWord(lpszPictureName,lpszObjectName,"Target_BackgroundColor",CO_WHITE);
}
else
{
SetPropBOOL(lpszPictureName,lpszObjectName,"Target_Permission",FALSE);
SetPropWord(lpszPictureName,lpszObjectName,"Target_ BackgroundColor ",CO_LTGRAY);
}
return TRUE;
```

   When one of these variables is changed, these will be read from the AS. The status of "@Level5" and "@Level6" also need to be verified.

   If all three criteria are met, set the "TARGET_PERMISSION“ property 'TRUE', the "TARGET_BackgoundColor“ property to "white".

Note: The script must also query the control "@Level" status of the direct connection.

4. Create a direct connection of the "Target_Permission“ property to the object that is actually subjected to a user rights verification, e.g. "Setpoint_AnalogValue", "Operator control enable“ property.

   The operator control enable property may not be interconnected to this object and must be set 'FALSE'.

   The background color must also be set to "gray" and is controlled with the operator control enable function by means of a script, or via direct connection using the "TARGET_BackgroundColor" property of the "Permission" object.

   The "Password" property is no longer used for the "Permission" object.

5. If necessary, add a direct connection of the "Level_Target“ property to a further "Permission" object.

2.3.24 "OpenNextFaceplate" Button

In an open faceplate, the "OpenNextFaceplate" object is used to call a further faceplate of an AS block from the same chart.

A typical application is the call of the corresponding INTERLOK block from a motor- or valve faceplate.

- The name of the block to be called is entered in the "block name“ property.
- The block type is entered in the usual format in the "Server name“ property, e.g. "PCS7 INTERLOK Control“ for the INTERLOK block.
- The variable name of the currently open faceplate is fetched by clicking on the button. Its block name is then truncated and the block name entered under "block name" is appended.
- The structured element "#Comment“, which exists in every block, is also appended and its existence in the file manager is verified.

If it does exist, the new variable name is written to the "tag name“ property and the faceplate is called by the "PCS7_OpenGroupDisplay_V6“ script.

A further script which is called when the "check tag" property is changed also verifies the existence of the variable. The button is only enabled for operator control if the variable is available in the file manager.

To ensure execution of the script in the "check_tag“ property when a new picture is selected, a valid structured element whose text is different to the default text of the property must be interconnected in order to detect changes of the property. The structured element "#Comment“ which exists in all blocks is used for this operation.

Another commonly used application lies in the implementation of multiple INTERLOK blocks for a single motor or valve.

In such situations, the same object can be installed in all the INTERLOK blocks, and a "1" is entered in the "block name" property for the cascaded call.

If the AS block name of the currently called faceplate variable (tag name) is not appended a number, e.g. its name is "L“, the AS block named "L1“ is called from the same chart in the next step, provided the corresponding variable name exists in the file manager.

If the AS block name of the currently called variable name of the faceplate is "L1“, a call of "L2“ is expected in the next step, etc.

This method basically allows the call of any number of INTERLOK blocks.

The configuration rule for this scenario is:

- An "OpenNextFaceplate" button with the block name "LOCK" is installed in a VALVE block.
- An "OpenNextFaceplate button with the block name "1" is installed in the INTERLOK block.
- INTERLOK blocks called by the faceplate type "VALVE" must be named "LOCK", "LOCK1", "LOCK2" etc. when using the method described above.

Authorizations:

The "OpenNextFaceplate" object is also assigned the "Processcontrolling_backup“ and "Higherprocesscontrolling_backup“ properties for setting operator control authorizations for the faceplate to be called. The usual default settings are here the authorization levels 5 and 6. You can transfer the authorizations entered in the source icon by means of direct connections of "@Level5" and "@Level6" to the "OpenNextFaceplate" object.

Direct connections:

@Level5/Authorization/Change → OpenNextFaceplate/Processcontrolling

@Level6/Authorization/Change → OpenNextFaceplate/HigherProcesscontrolling

2.4 Scripts

Script lists

The scripts listed below are installed onto the "\Siemens\WinCC\aplib\FacePlateDesigner_V6" or "\Siemens\WinCC\aplib\FacePlateDesigner" directory and must **not** be copied to the GraCS folder of the project directory. The scripts are not project-specific, but rather specific to a computer.

Script name	Function
PCS7_OpenGroupDisplay_V6.Fct	Opens a "Group display" faceplate
PCS7_OpenGroupDisplay_I_V6.Fct	Opens a corresponding INTERLOK block for drives via a "Right"-click
PCS7_UpdateGroupTagname_V6.fct	A script called when the "tag name" property of the "@Faceplate" object is changed. The data are written to the "tag name“ property when the faceplate is called.
PCS7_OpenLoopDisplay_V6.Fct	Opens the loop display; call of the "Loop" button in "@PG_%Type%.pdl"
PCS7_UpdateLoopTagname_V6.Fct	A script called when the "tag name" property of the "@Faceplate" object is changed in the loop picture. The data are written to the "tag name“ property when the faceplate is called.
PCS7_CheckPermission.fct	Verifies authorizations
PCS7_UpdatePermission_V6.Fct	Called when the "@CurrentUser" changes and when a view is selected in the pictures "@PG_%Type%.pdl" and "@PG_%Type%.pdl"
PCS7_ChangeView.fct	Calls another view in the faceplate; call from "@PG_%Type%_Viewlist.pdl"
PCS7_OperationLog_V6.fct	Operator control log for WinCC
PCS7_OpenCheckbox_V6.Fct	Called from the basic element "CHECKBOX_L1/R1". Opens the operator control picture "@PCS7_BedCheck.pdl"
PCS7_Check_OK_V6.fct	Script called via "OK" button in "@PCS7_BedCheck.pdl"
PCS7_OpenComboBox_V6.fct	Opens the operator control picture "@PCS7_BedCombo.pdl" from the basic element "PCS7_COMBOBOX1"
PCS7_Combo_OK_V6.fct	Script called via "OK" button in "@PCS7_BedCombo.pdl"
PCS7_Open3ComboBox_V6.fct	Opens the operator control picture "@PCS7_3BedCombo.pdl" from the basic element "PCS7_3COMBOBOX1"
PCS7_3Combo_OK_V6.fct	Script called via "OK" button in "@PCS7_3BedCombo.pdl"
PCS7_OpenInputBoxBin_V6.fct	Opens the operator control picture "@PCS7_BedBinaer.pdl" from the basic elements "PCS7_BinOp" and "PCS7_ButtonBit"
PCS7_Binary_OK_V6.fct	Script called via the "OK" button in "@PCS7_BedBinaer.pdl"
PCS7_2Stati_Variable_Changed_V6.fct	Called in the basic element "PCS7_Status_2_Alternative“
PCS7_1vnStati_Variable_Changed_V6.fct	Called in the basic element "PCS7_Status_1_v_n“
PCS7_OpenInputBoxAnalog_V6.fct	Opens the operator control picture "@PCS7_BedAnalog.pdl" or "@PCS7_BedAnalog_NL.pdl" from the basic element "PCS7_AnalogValue"

Script name	Function
PCS7_OpenGroupDisplay_V6.Fct	Opens a "Group display" faceplate
PCS7_Analog_OK_V6.fct	Script called via "OK" button in "@PCS7_BedAnalog.pdl" or "@PCS7_BedAnalog_NL.pdl"
PCS7_AnalogPercent_V6 .fct	Script called for incremental operator control in "@PCS7_BedAnalog.pdl"
PCS7_UpdateBarLimits_V6.Fct	Script called in the basic element "PCS7_BarLimits"
PCS7_UpdateBar_V6.Fct	Script called in the basic elements "PCS7_BarStandard1" and "PCS7_BarStandard2"
PCS7_Format_V6	Script for the transfer of number formats
PCS7_Trend_V6.Fct	Script for the display of the trend in a faceplate

The "PCS7_ChangeView.fct“ and "PCS7_CheckPermission.fct“ scripts are migrated from V5 without changes and thus do not carry a V6 in their name.

2.5 Bitmaps

The bitmaps are installed to the "...\Siemens\WinCC\options\pdl\FaceplateDesigner_V6" directory.

The bitmaps are copied to the "GraCS" folder of the project directory during the execution of the OS project editor.

The bitmaps are loaded dynamically into the corresponding selected pictures.

File name of the bitmap	Icon display	Used in	Directory
@FP_PopUpIcon.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_AlarmCrossed.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_AlarmDisabled.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_AlarmEnabled.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_NotOccupied.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_Occupied.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_OpenLoop.bmp		Faceplate, global	pdl\FaceplateDesigner
@PCS7_Lock.bmp		Motor / valve faceplate	pdl\FaceplateDesigner
@PCS7_UnLock.bmp		Motor / valve faceplate	pdl\FaceplateDesigner
@CollectValue_S.emf		Block icon, global ASD	
@CollectValue_F.emf		Block icon, global ASD	
@CollectValue_transparent.		Block icon, global ASD	
@CollectValue_empty.emf		Block icon, global ASD	
@Ctrl_Manual.emf		Controller/block icon	
@Ctrl_intern.emf		Controller/block icon	
@Ctrl_extern.emf		Controller/block icon	
@Ctrl_Auto.emf		Controller/block icon	
@Ctrl_Track.emf		Controller/block icon	
@off.emf		Block icon OPD	pdl\Base_Data_Poo
@on.emf		Block icon OPD	pdl\Base_Data_Poo
@Auto.emf		Block icon in general	pdl\Base_Data_Poo
@Manual.emf		Block icon in general	pdl\Base_Data_Poo
MOTOR_IS_OFF.emf		Motor faceplate and block icon	pdl\FaceplateDesigner
MOTOR_Error.emf		Motor faceplate and block icon	pdl\FaceplateDesigner
MOTOR_IS_ON		Motor faceplate and block icon	pdl\FaceplateDesigner

File name of the bitmap	Icon display	Used in	Directory
MOTOR_OFF.emf		Motor faceplate and block icon	pdl\FaceplateDesigner
MOTOR_ON.emf		Motor faceplate and block icon	pdl\FaceplateDesigner
VAZ_H.emf		Valve faceplate and block icon	pdl\FaceplateDesigner
VAZ_H_CLOSE.emf		Valve faceplate and block icon	pdl\FaceplateDesigner
VAZ_H_OPEN.emf		Valve faceplate and block icon	pdl\FaceplateDesigner
Valve_NL.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
Valve_L.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHO_closed.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHO_opened.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHO_Error.emf		Valve block icon (interlock)	
VHO_undef.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHZ_closed.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHZ_opened.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VHZ_undef.emf		Valve block icon (interlock)	pdl\FaceplateDesigner
VVE_closed.emf		Valve block icon	pdl\FaceplateDesigner
VVE_opened.emf		Valve block icon	pdl\FaceplateDesigner
VVE_Error.emf		Valve block icon	pdl\FaceplateDesigner
VVE_undef.emf		Valve block icon	pdl\FaceplateDesigner
VVT_closed.emf		Valve block icon	pdl\FaceplateDesigner
VVT_opened.emf		Valve block icon	pdl\FaceplateDesigner
VVT_undef.emf		Valve block icon	pdl\FaceplateDesigner

2.6 Pictures

The pictures listed below are installed to the "...\Siemens\WinCC\options\pdl\FaceplateDesigner_V6" directory.

These are copied to the "GraCS" folder of the project directory during the execution of the OS project editor.

Pictures	
@PCS7Elements.Pdl	Template picture for basic elements
@@PCS7Typicals.pdl	Template picture of block icons for the PH, supplement block icons.
@Template.pdl	Template picture of block icons for Graphics Object Update; Different to "@@PCS7Typicals.pdl" only in the "type“ property.
@PCS7_BedAnalog.pdl	Picture for operator control of analog values
@PCS7_BedAnalog_NL.pdl	Picture for operator control of analog values, without limits, bars or incremental process control.
@PCS7_BedBinaer.pdl	Operator control picture for binary, single-button operation, only acknowledgment (e.g. Open/Close/Manual/Auto for valve/motor).
@PCS7_BedCheck.pdl	Picture for operator control of binary two-button controls (e.g. set Alarm/Warning)
@PCS7_BedKombo.pdl	Picture for operator control of a binary two-button combo box (e.g. manual/auto mode for controller)
@PCS7_3BedKombo.pdl	Picture for operator control of binary three-point combo boxes (e.g. manual/auto mode for controller)
@PCS7_AnalogInputwithLimits.pdl	Picture for operator control of analog values (not used, but included in delivery for reasons of downward compatibility to V5.1)
@PCS7_BinaryInput1of2.pdl	Picture for operator control of binary values (not used, but included in delivery for reasons of downward compatibility to V5.1)
@PCS7_ALARM.pdl	Display of the message view in faceplates with alarm
@PCS7_BATCH.pdl	Display of the batch view in faceplates
@PCS7_TREND.pdl	Display of the trend view in analog faceplates
@PG_%Type%.pdl	Template picture for prototype picture faceplate
@PG_%Type%_%View%.pdl	Template picture for sublevel views in the windows
@PG_%Type%_VIEWLIST.pdl	Template picture for the view selection menu
@PG_%Type%_OverView.pdl	Template picture for the overview
@PL_%Type%.pdl	Template picture for the loop display

2.7 Faceplates

General Notes

For information on which parameters of the AS block instance can be used to visualize the various display blocks in the faceplate, refer to the offline dialog of the Graphics Designer.

Block comment

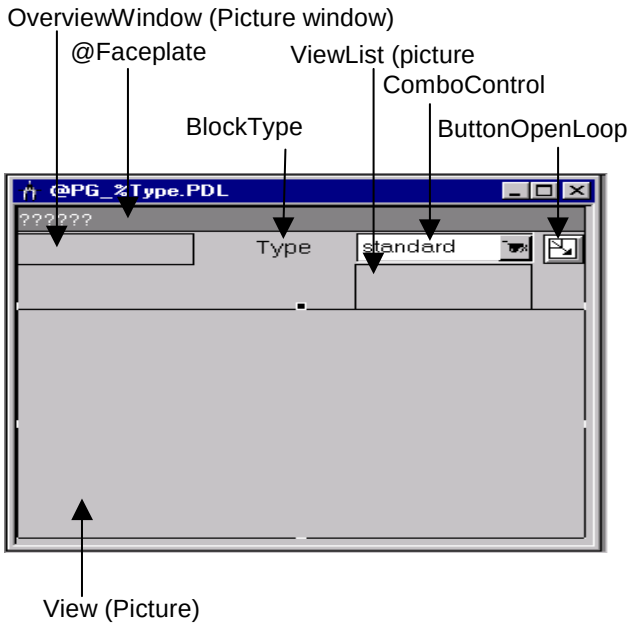
The CFC block comment is displayed in the faceplate in the form of a short information text (tool tip) showing the variable name.

This method ensures that the description of the technological function is always available in the faceplate.

2.7.1 Basic Data of the Picture Templates

2.7.1.1 @PG_%Type%.pdl

The "@PG_%Type%.pdl" template is available for configuring projects.



The various elements are assigned the following functions:

Overview Window

Picture window for displaying the alarm status, batch status, instance-specific message acknowledgment and for enabling messages of the faceplate. A group display is usually shown here. If the faceplate is used for a multiple instance block, it may also show several group displays. The display is defined in the configuration of the "@PG_<Type>_Overview.pdl“ picture. The same picture is also shown in the loop view of the faceplate.

@Faceplate

Displays the block instance or the variable of the calling object and stores additional information for the faceplate, e.g. First View, name of the batch variable, Current User etc.

Trend Functions

User object with I/O field, for storing data of the trend function "Trendpage". This object is hidden in online mode.

Block Type

Name of the faceplate type. This type name forms part of the name of all views belonging to the faceplate. This object is hidden in online mode.

View List

Picture window for the display and selection of available views.

Combo Control

Selection and display element for the various views. This element always shows the name of the current view.

Button Open Loop

Object for selecting the loop picture. The Faceplate Designer automatically hides this object, if generation or update of the loop view is not selected.

View

Picture window for the display of the various faceplate views.

Operation Window

Picture window for the display of analog operations.

Combo Window

Picture window for the display of binary operations.

2.7.1.2 @PG_%TYPE%

Picture object @PG_%TYPE%

Geometry/Pos	X=0, y = 0
Geometry/Dim	Width = 320, Height = 260

Picture window "View"

Geometry/Pos	X=1, y = 47
Geometry/Dim	Width = 320, Height = 214

User object @Faceplate

The user object "@Faceplate" consists of the following elements:

Property in the user object	Element	Type	Default
Geometry/Pos	X=0, y = 0		
Geometry/Dim	Width = 320, Height = 20		
Tagname	Tagname	Stat. text	none
Tag	Tag	Stat. text	Text = MKZ
FirstView	FirstView	I/O field	Output value is set by the Faceplate Designer
CurrentUser	CurrentUser	I/O field	Output value = Interconnection to internal variable @Current User
Bname	VarBatchname	I/O field	Output value = .BA_NA
BATCH_ID	VarBatchID	I/O field	Output value = .BA_ID
STEP_NO	VarBatch Stepnumber	I/O field	Output value = .STEP_NO
STEP_N1	VarBatch Stepnumber _N1	I/O field	Output value = .STEP_N1
Areaname	Areaname	I/O field	Output value = .#areaname
Processcontrolling_backup	POP	I/O field	Authorization = Processcontrolling
HigherProcess controlling_backup	HIPOP	I/O field	Authorization = Higher Processcontrolling
MULTI_INSTANCE	HIPOP.Hidden Input	I/O field	True = MultiInstanceFaceplate

2.7.1.3 @PG_%Type%_%View%.pdl

Picture object @PG_%Type%_%View%

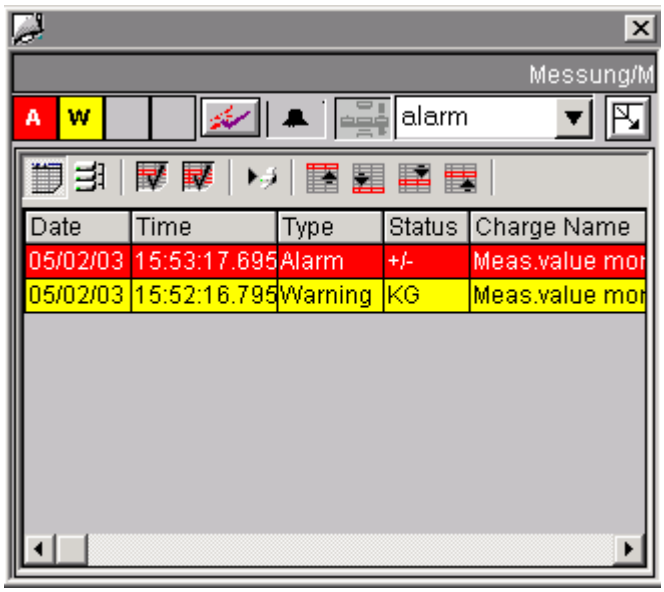
Geometry/Pos	X=0, y = 0
Geometry/Dim	Width = 320, Height = 214

Rectangle@Frame

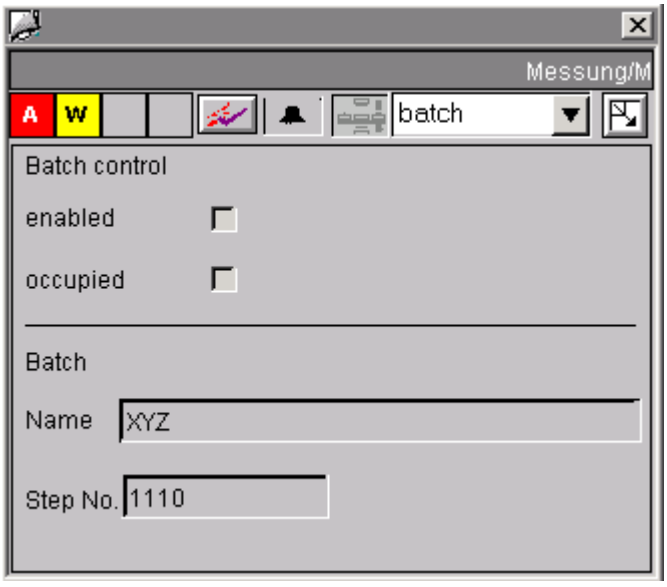
Geometry/Pos	X=1, y = 50
Geometry/Dim	Width = 320, Height = 214

2.7.2 Global Views

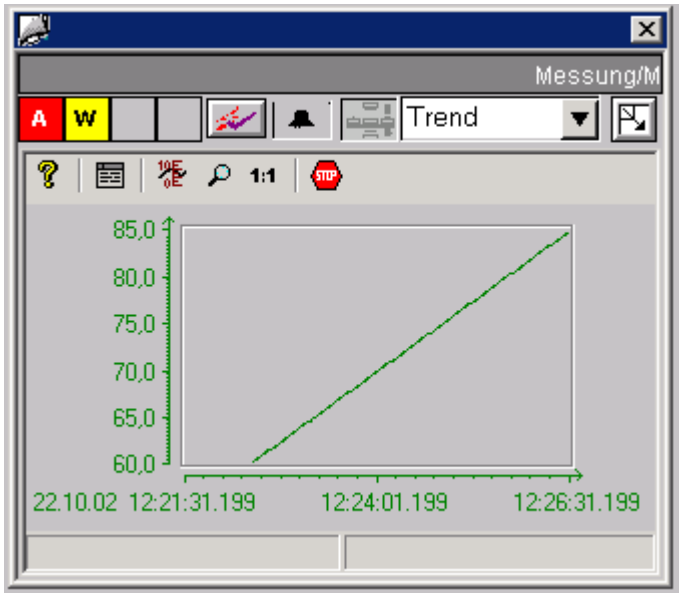
2.7.2.1 Message View



2.7.2.2 Batch View



2.7.2.3 Trend View



See also Chapter 2.1.7, "Configuring the Trend View".

2.7.3 CTRL_PID

In the following, the CTRL_PID faceplate with its "standard", "maintenance", "parameter" and "limits" views is described as an example for PCS 7 faceplates.

For more information about all faceplates, refer to the online help for the PCS 7 Faceplates.

2.7.3.1 CTRL_PID: Standard View

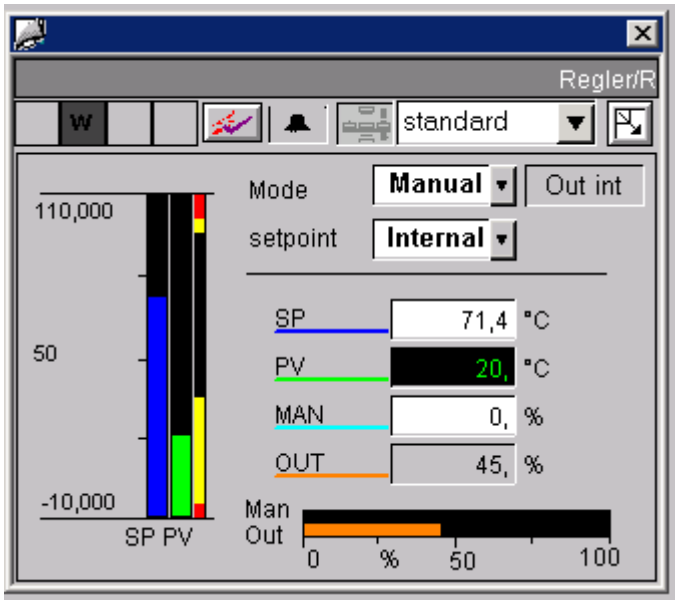
Faceplate standard view as of V6.0

All analog displays are created by means of the "AdvancedAnalogDisplay". The number format is set via the block icon ("Format_InputValue“ and "Format_OutputValue“ properties). See section 2.1.6, "Configuring Number Formats".

View 2 "Permission“ also has objects for the input of setpoints and manipulated variables, since operator authorizations for these variables depend upon various factors. See also Chapter 2.3.23, Basic Elements, Permission Object.

The "Permission_Setpoint“ object evaluates the WinCC authorization levels, as well as the "Q_SP_OP = TRUE“ parameter..

The "Permission_Manual“ object evaluates the WinCC authorization levels, as well as the "QLMNOP = TRUE“ parameter.



The PID tuner is operated in the parameter view (Tuning On/Off).

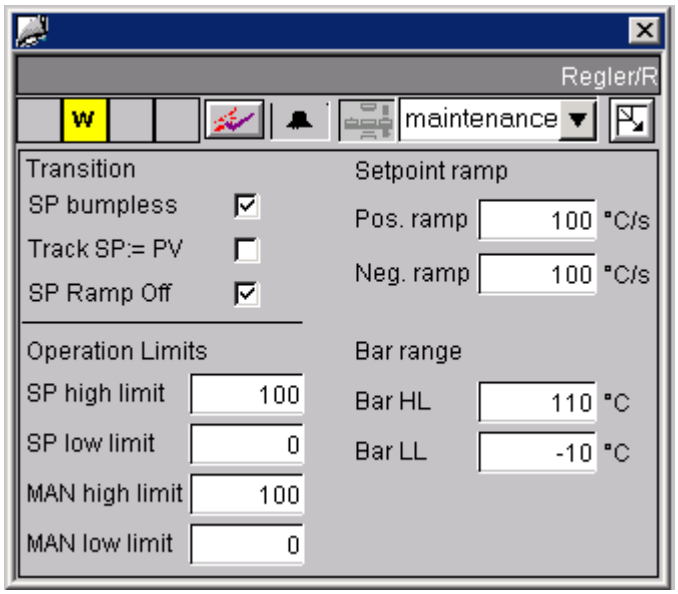
When tuning is active, the standard view displays a combo box that is called via the operating mode combo box "Manual/Auto", which can also be used to switch off tuning in the standard view. All other operations of the controller are locked when "Tuning On“ is set.

Order and assignment of direct connections to operator controlled objects

@Level5	->	**Operator control enable**
Manual_COMBOBOX	->	Operator control enable
External_COMBOBOX	->	Operator control enable
Permission_Setpoint	->	Level_Source -> Level_Target
Permission_Manual	->	Level_Source
Permission_Setpoint	->	**Target_Operator control enable**
Setpoint_AnalogValue	->	Operator control enable
Permission_Manual	->	**Target_Operator control enable**
Manual_AnalogValue	->	Operator control enable
Format	->	**Format_InputValue**
Setpoint_AnalogValue	->	Format
ProcessValue_AnalogValue	->	Format
Format	->	**Format_OutputValue**
Manual_AnalogValue	->	Format
Output_AnalogValue	->	Format

2.7.3.2 CTRL_PID: Maintenance View

The "Permission_SP_Bumpless“ object evaluates the WinCC authorization levels, as well as the "OPTI_EN = FALSE“ parameter.



Order and assignment of direct connections to operator controlled objects

@Level6	->	**Operator control enable**
Permission_SP_Bumpless	->	Level_Source
Permission_SP_Bumpless	->	**Target_Operator control enable**
Bumbless_CHECKBOX_L	->	Operator control enable
SP_TRK_ON_CHECKBOX_L	->	Operator control enable
SPRAMP_OFF_CHECKBOX_L	->	Operator control enable
SPHighLimit_AnalogValue	->	Operator control enable
SPLowLimit_AnalogValue	->	Operator control enable
ManHighLimit_AnalogValue	->	Operator control enable
ManLowLimit_AnalogValue	->	Operator control enable
SPURLM_AnalogValue	->	Operator control enable
SPDRLM_AnalogValue	->	Operator control enable
MO_PVHR_AnalogValue	->	Operator control enable
MO_PVLR_AnalogValue	->	Operator control enable

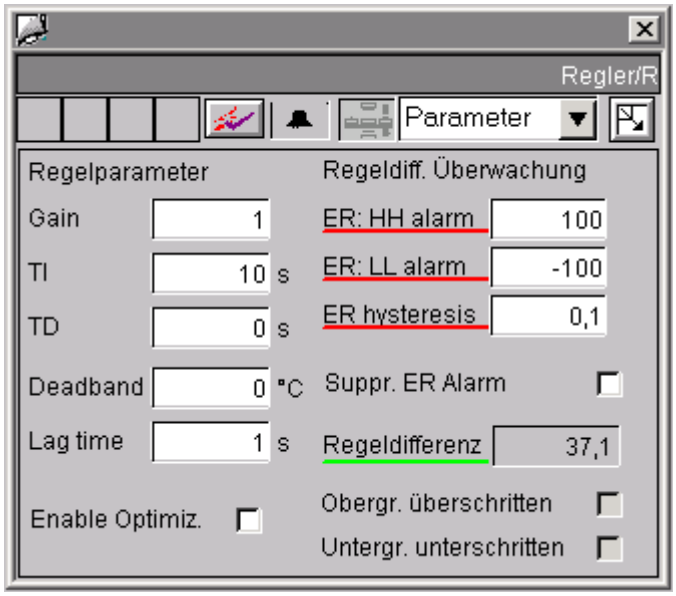
Permission_SP_Bumpless	->	**Target_BackgroundColor**
SPHighLimit_AnalogValue	->	Background color value
SPLowLimit_AnalogValue	->	Background color value
ManHighLimit_AnalogValue	->	Background color value
ManLowLimit_AnalogValue	->	Background color value
SPURLM_AnalogValue	->	Background color value
SPDRLM_AnalogValue	->	Background color value
MO_PVHR_AnalogValue	->	Background color value
MO_PVLR_AnalogValue	->	Background color value

2.7.3.3 CTRL_PID: Parameter View

The process value "Error signal_AnalogValue“ is set via the "AdvancedAnalogDisplay", the number format is set via the block icon ("Format_InputValue“ property).

All other analog displays show the conventional "Floating-point format“ I/O field.

The "Permission_Gain“ object evaluates the WinCC authorization levels, as well as the P "OPTI_EN = FALSE“ parameter.



Order and assignment of direct connections to operator controlled objects

@Level6	->	**Operator control enable**
Permission_Gain	->	Level_Source
OPTI_EN_CHECKBOX_L	->	Operator control enable
Permission_Gain	->	**Target_Operator control enable**
Gain_AnalogValue	->	Operator control enable
TN_AnalogValue	->	Operator control enable
TV_AnalogValue	->	Operator control enable
DEADB_W_AnalogValue	->	Operator control enable
TM_LAG_AnalogValue	->	Operator control enable
ERH_ALM_AnalogValue	->	Operator control enable
ERL_ALM_AnalogValue	->	Operator control enable
ER_HYS_AnalogValue3	->	Operator control enable
M_SUP_ER_CHECKBOX_L	->	Operator control enable

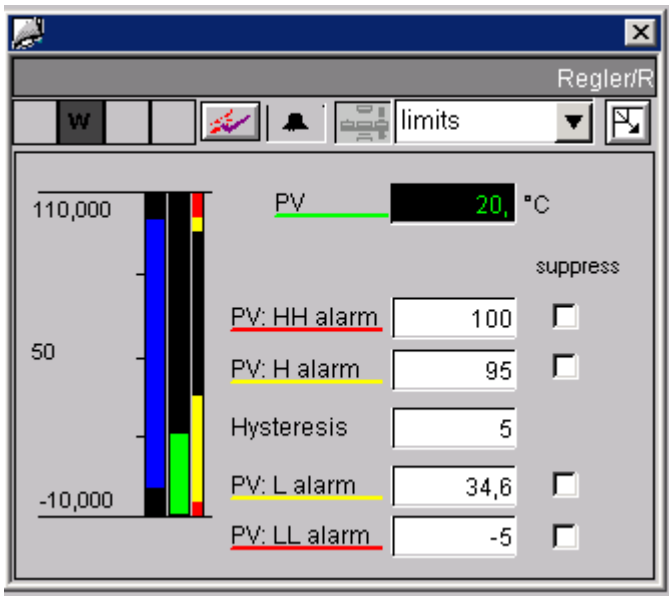
Permission_Gain	->	**Target_ BackgroundColor**
Gain_AnalogValue	->	Background color_value
TN_AnalogValue	->	Background color_value
TV_AnalogValue	->	Background color_value
DEADB_W_AnalogValue	->	Background color_value
TM_LAG_AnalogValue	->	Background color_value
ERH_ALM_AnalogValue	->	Background color_value
ERL_ALM_AnalogValue	->	Background color_value
ER_HYS_AnalogValue3	->	Background color_value
Format	->	**Format_InputValue**
Error signal_AnalogValue	->	Format

2.7.3.4 CTRL_PID: Limits View

The process value "ProcessValue_AnalogValue“ is created via the "AdvancedAnalogDisplay". The number format is set via the block icon ("Format_InputValue“ property).

All other analog displays show the conventional "Floating-point format“ I/O field.

The "Permission_AlarmHigh_AnalogValue“ object evaluates the WinCC authorization levels, as well as the "OPTI_EN = FALSE“ parameter.



This setpoint bar graph shows the setpoint control limits, with reference to the bar graph limits.

Setpoint control limits are set in the maintenance view.

Order and assignment of direct connections to operator controlled objects

@Level6	->	**Operator control enable**
Permission_AlarmHigh_AnalogValue	->	Level_Source
Permission_AlarmHigh_AnalogValue	->	**Target_Operator control enable**
AlarmHigh_AnalogValue	->	Operator control enable
WarningHigh_AnalogValue	->	Operator control enable
Hysterese_AnalogValue	->	Operator control enable
WarningLow_AnalogValue	->	Operator control enable
AlarmLow_AnalogValue	->	Operator control enable
AlarmHigh_CHECKBOX_R	->	Operator control enable
WarningHigh_CHECKBOX_R	->	Operator control enable
WarningLow_CHECKBOX_R	->	Operator control enable
AlarmLow_CHECKBOX_R	->	Operator control enable
Permission_AlarmHigh_AnalogValue	->	**Target_BackgroundColor**
AlarmHigh_AnalogValue	->	BackColor_value
WarningHigh_AnalogValue	->	BackColor_value
Hysterese_AnalogValue	->	H BackColor_value
WarningLow_AnalogValue	->	BackColor_value
AlarmLow_AnalogValue	->	BackColor_value
Format	->	**Format_InputValue**
ProcessValue_AnalogValue	->	Format

2.8 Block Icons

Note: There are no provisions made for process control of block icons. All process controls derive from the faceplates.

2.8.1 Picture Templates @@PCS7Typicals.pdl and @Template.pdl

The previously supplied block icons (V5.x) in the pictures @@**PCS7Typicals.pdl** and @**Template.pdl** can open all faceplate versions (OCX or faceplate V5.1 / V5.2 / V6.0) in the corresponding prototype picture.

However, these new functions are available only for use with the new block icons.

The new block icons are stored in the pictures @@**PCS7Typicals.pdl** and @**Template.pdl**.

The picture @@**PCS7Typicals.pdl** is used for the automatic creation of block icons from the PH.

When the PH contains a picture for which the "Derive Block Icons from PH“ option is set, the block icons for this picture are created in this folder for all OS-relevant CFC blocks in the charts of this hierarchy folder and, depending of the settings, the nested folders, if:

- You use the "Create/Modify Block Icons " menu command in the PH

  or

- You have enabled the corresponding option in the wizard when the function "Compile OS " was executed.

The following applies here:

For a CFC block instance with the symbolic type name CTRL_PID, a block icon is copied into this picture from the @@**PCS7Typicals.pdl** picture, and its "type“ property is defined by the string "@CTRL_PID/1“.

To modify the picture "@@PCS7Typicals.pdl", you need to save a copy of it under the name "@PCS7Typicals.pdl" and then edit the copy. The picture "@PCS7Typicals.pdl" is automatically derived from the PH, provided it exists in the project.

Note

All block icons in the pictures which also exist in "@@PCS7Typicals.pdl" and have not been generated via the PH will be deleted during automatic generation. Hence, the "@Template.pdl" picture must always be used as a template for the block icons for manual configurations and for post-processing such pictures, since the "type" property is preset differently.

As of V6, this reference can be configured on a CFC block instance, and there will be no mandatory naming convention at the "type“ property. In addition, you can generate multiple different block icons for one block type in the ES.

Example: The symbolic name "XXX" is entered at a CTRL_PID instance. A reference is created in the picture @@**PCS7Typicals.pdl** to a block icon that contains the string "@CTRL_PID/XXX“ in its "type" property.

The @**Template.pdl** picture is primarily used as template for manual configuration of block icons in the WinCC pictures. The difference between the block icons in these two pictures is found in the "**type**“ property, which may not be modified in the picture "@@PCS7Typicals.pdl" (e.g. the naming convention @MEAS_MON/1), since this is the reference that determines which objects are generated via the PH and deleted.

This property may not be changed in @**Template.pdl**.

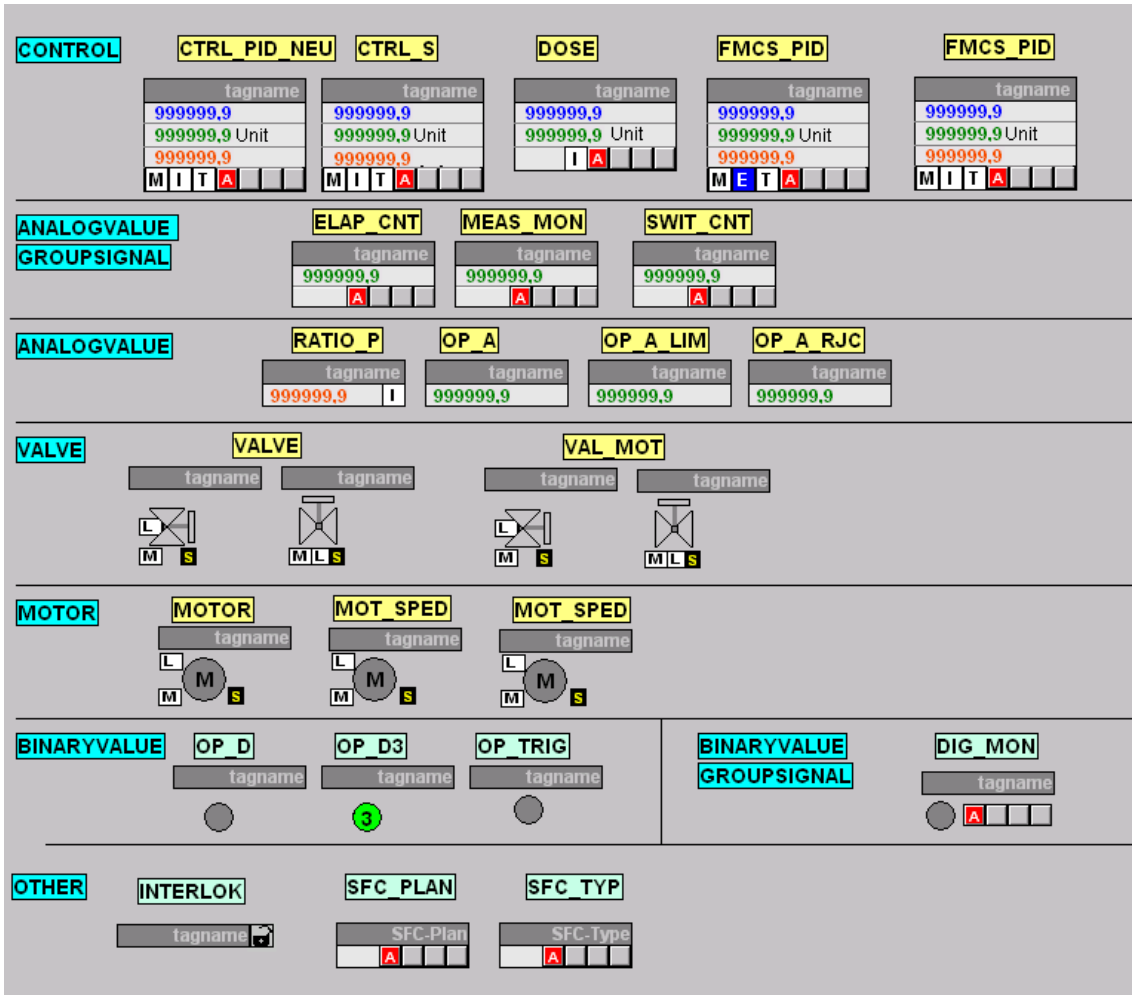
You should never assign this property the name of block icon which exist in @@PCS7Typicals, since you will run the risk that the block icons copied from this template are deleted in the pictures you generate via the PH.

If there is a reason to modify existing icons, it is advisable not to modify the "@Template.pdl" picture itself, but rather the copy thereof which you have saved under a different name. Otherwise, the OS Project Editor (previously the Split Screen Wizard) will reset the picture.

Both the pictures @@**PCS7Typicals.pdl** and @**Template.pdl** can be used for the "**Update Picture Objects**“.

Here, the "type“ property also forms the reference to objects to be replaced.

2.8.2 Block Icon in the Picture @@PCS7_Typicals



2.8.3 Block Icon Properties

2.8.3.1 General Properties

The following properties of the block icons of the "@@PCS7Typicals" picture should never be modified:

- Geometry/Width
- Geometry/Height
- Other/Operator control enable
- Other/Password
- Other/Display
- General/Server name
- Styles/GroupRelevant (only for blocks with Alarm_8P messages)

The following properties exist in all block icons:

Properties	Element and property in the user object	Object	Description
Other/Processcontrolling_backup	POP.authorization	I/O field	Instance-spec.authorization, Default = 5
Other/ HigherProcesscontrolling_backup	HIPOP.authorization	I/O field	Instance-spec.authorization, Default = 6
General/tag	NameOfTag.OutputValue	I/O field	Text displayed in the icon
General/type	Type. OutputValue	I/O field	Reference for the generation of icons from the PH and for Wizards
General/tagname	Tagname.OutputValue	I/O field	Actual variable name that is passed to the variable prefixes of the picture windows
General/Servername	Servername.OutputValue	I/O field	Block type or faceplate type
General/Version	Version.OutputValue	I/O field	Version number
Styles/View_Tag	NameOfTag.Display Rectangle17.Display (if it exists)	Rectangle I/O field	Can be used to hide the variable name
MouseClick left	PCS7_OpenGroupDisplay_V6 (lpszPictureName, lpszObjectName)		Calls the faceplate

2.8.3.2 CTRL_PID

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 110 / Height = 77		
General/ UnitPV	UnitPV.Text	Stat.Text	Display: Unit PV
General/ Unit_MAN_OP	Unit_MAN_OP.Text	Stat.Text	Display: Unit MAN_OP
Links/ CollectValue	GroupDisplay.CollectValue	GroupDisplay.	.EventState
Links/ SetpointValue	SetpointValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Setpoint
Links/ ProcessValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Links/ OutputValue	OutputValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Manipulated variable
Links/ LMN_SEL	Tracking_AdvancedStatusDisplay.Status	AdvancedStatusDis.	Display: Manipulated value tracking
Links/ Mode_MAN_AUT	Manual_AdvancedStatusDisplay.Status	AdvancedStatusDis.	Display: Manual/Auto
Links/ Mode_INT_EXT	External_AdvancedStatusDisplay.Status	AdvancedStatusDis.	Display: External/Internal
Styles/ ReturnPath	TrendFunctions2 .OutputValue	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ StandardTrend	TrendFunktions2 .CharacterSetSize	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles / Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format SetpointValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis. AdvancedAnalogDis.	Number formatting for process value and setpoint
Styles/ Format_OutputValue	OutputValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formatting of the numbers of manipulated variables
Styles/ Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.3 CTRL_S

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 110 / Height = 77		
General/UnitPV	Unit.Text / .PV_IN#unit	Stat. text	Display: Unit PV
General/ Unit_MAN_OP	Unit.Text / .MAN_OP#unit	Stat. text	Display: Unit manipulated variable
Links/CollectValue	GroupDisplay.CollectValue/	GroupDisplay	.EventState
Links/SetpointValue	SetpointValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Setpoint
Links/ProcessValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Links/OutputValue	OutputValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Manipulated variable
Links/ Mode_MAN_AUT	Manual_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: Manual/Auto
Links/ Mode_INT_EXT	External_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: External/Internal
Links/LMN_SEL	Tracking_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: Manipulated variable correction
Links/QLMNR_ON	OutputValue_Advanced AnalogDisplay.Display Unit_MAN_OP.Display	AdvancedAnalogDis. Stat. text	See below for the description
Links/QLMNUP	LMNUP_StatusDisplay	Stat. text	Display:QLMNUP
Links/QLMNDN	LMNDN_StatusDisplay	Stat. text	Display:QLMNDN
Styles/ReturnPath	TrendFunctions2 .OutputValue	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/StandardTrend	TrendFunctions2 .CharacterSetSize	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format SetpointValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis. AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	OutputValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the manipulated variable numbers
Styles/Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

The CTRL_S block icon differs from that of CTRL PID in as far that it displays the binary control signals QLMNUP and QLMNDN in applications without position feedback (LMNR_ON = 0), instead of the manipulated variable.

The visibility of these texts is also controlled by scripts, which are called upon changes in the QLMNUP and QLMNDN properties.

Note: The "OutputValue_AdvancedAnalogDisplay“ and "Unit_MAN_OP“ objects must always be brought to the foreground in the user object, in order to ensure proper functioning of the visualization control.

2.8.3.4 DOSE

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 110 / Height = 63		
General/ UnitPV	UnitPV.Text	Stat. Text	Display: Unit PV
Links/ CollectValue	GroupDisplay.CollectValue	GroupDisplay	.EventState
Links/ ProcessValue	ProcessValue_AdvancedAnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Links/ SetpointValue	SetpointValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Setpoint
Links/ SetpointExtern	External_ AdvancedStatusDisplay.Status SetpointExternValue_ Advanced AnalogDisplay .Display	AdvancedStatusDis AdvancedAnalogDis	.QSPEXTON See below for a description
Links/ ValueSetpointExtern	SetpointExternValue_ AdvancedAnalogDisplay.Value	AdvancedAnalogDis	Displayed with .QSPEXTON via the setpoint
Styles/ ReturnPath	TrendFunctions2 .OutputValue	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ StandardTrend	TrendFunctions2 .CharacterSetSize	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format SetpointValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis. AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	OutputValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the manipulated variable numbers
Styles/ Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

Since the DOSE block does not contain a parameter to represents the effective setpoint, the setpoint display is shown according to the setting in QSPEXTON.

QSPEXTON = 0 → "SetpointValue_AdvancedAnalogDisplay" is displayed

QSPEXTON = 1 → "SetpointExternValue_ AdvancedAnalogDisplay " is displayed

2.8.3.5 FMCS_PID / FMT_PID

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 110 / Height = 77		
General/UnitPV	Unit.Text / .PV#unit	Stat. Text	Display: Unit PV
General/ Unit_MAN_OP	Unit.Text / .LMN#unit	Stat. Text	Display: Unit manipulated variable
Links/CollectValue	GroupDisplay.CollectValue/	GroupDisplay	.EventState
Links/ SetpointValue	SetpointValue_Advanced AnalogDisplay.Value/.SP	AdvancedAnalogDis.	Display: Setpoint
Links/ ProcessValue	ProcessValue_Advanced AnalogDisplay.Value/.PV	AdvancedAnalogDis.	Display: Process value
Links/ OutputValue	OutputValue_ Advanced AnalogDisplay.Value/.LMN	AdvancedAnalogDis.	Display: Manipulated variable
Links/ Tracking	Tracking_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: Tracking LMN
Links/ Mode_MAN_AUT	Manual_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: Manual/Auto
Links/ Mode_INT_EXT	External_AdvancedStatus Display.Status	AdvancedStatusDis.	Display: External/Internal
Styles/ ReturnPath	TrendFunctions2 .OutputValue	IO field	See section 2.1.7, "Configuring the Trend View"
Styles/ StandardTrend	TrendFunctions2 .CharacterSetSize	IO field	See section 2.1.7, "Configuring the Trend View"
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format SetpointValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis. AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	OutputValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the manipulated variable numbers
Styles/Format_xx	Format_xx.OutputValue	IO field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.6 ELAP_CNT

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 97 / Height = 45		
General/Unit	Unit.Text	Stat. text	Display: Unit
Links/CollectValue	GroupDisplay.CollectValue	GroupDisplay	.EventState
Links/Output_Value	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	.HOURS Display max. 7 digits
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	Format_OutputValue. OutputValue	I/O field.	Formats the manipulated variable numbers
Styles/Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.7 MEAS_MON

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 97 / Height = 45		
General/Unit	Unit.Text	Stat. text	Display: Unit
Links/CollectValue	GroupDisplay.CollectValue	GroupDisplay	.EventState
Links/ OutputValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Styles/ ReturnPath	TrendFunctions2 .Output value	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ StandardTrend	TrendFunctions2 .CharacterSetSize	I/O field	See section 2.1.7, "Configuring the Trend View"
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	Format_OutputValue .OutputValue	I/O field	Formats the manipulated variable numbers
Styles/Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.8 SWIT_CNT

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 97 / Height = 45		
General/Unit	Unit.Text	Stat. text	.V#UNIT
Links/CollectValue	GroupDisplay.CollectValue	GroupDisplay	.EventState
Links/ OutputValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process Value
Styles/ Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	Format_OutputValue .OutputValue	I/O field	Formats the manipulated variable numbers
Styles/Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.9 RATIO_P

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 97 / Height = 32		
General / Unit	Unit.Text	Stat. text	Display: Unit
Links / OutputValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Links / Mode_INT_EXT	External_ AdvancedStatusDisplay.Status	AdvancedStatusDis.	Display: External/Internal
Styles / Format_InputValue	ProcessValue_advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles/ Format_OutputValue	Format_OutputValue .OutputValue	I/O field.	Formats the manipulated variable numbers
Styles / Format_xx	Format_xx.output value	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.10 OP_A

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 97 / Height = 32		
General / Unit	Unit.Text	Stat. text	Display: Unit
Links / OutputValue	ProcessValue_Advanced AnalogDisplay.Value	AdvancedAnalogDis.	Display: Process value
Styles / Format_InputValue	ProcessValue_Advanced AnalogDisplay.Format	AdvancedAnalogDis.	Formats the process value and setpoint numbers
Styles / Format_OutputValue	Format_OutputValue .OutputValue	I/O field.	Formats the manipulated variable numbers
Styles / Format_xx	Format_xx.OutputValue	I/O field	Further format, see Chapter 2.1.6, "Configuring Number Formats"

2.8.3.11 OP_A_LIM

Properties and display as OP_A. See section 2.8.3.10, OP_A

2.8.3.12 OP__A_RJC

Properties and display as OP_A. See section 2.8.3.10, OP_A

2.8.3.13 VALVE

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 67		
Links/CollectValue	GroupDisplay_withASD .CollectValue	AdvancedStatusDis.	.EventState
Links/QMAN_AUT	Mode.Status1	AdvancedStatusDis.	Display: Auto/Manual
Links/V_LOCK	Interlock.Status1	AdvancedStatusDis.	Display: Lock
Links/QOPENED	Valve_Status.Status1	AdvancedStatusDis.	Display: Valve
Links/QCLOSED	Valve_Status.Status2	AdvancedStatusDis.	Display: Valve
Links/QOPENING	Valve_Status.Status3	AdvancedStatusDis.	Display: Valve
Links/QCLOSING	Valve_Status.Status4	AdvancedStatusDis.	Display: Valve

Left-click opens the VALVE faceplate, right-click calls the corresponding INTERLOK faceplate.

The name of the INTERLOK block is stored as script transfer parameter, see Chapter 2.4 "Scripts"

The default block name is "L“. The INTERLOK block and the VALVE block must be placed in the same CFC chart.

2.8.3.14 VAL_MOT

Properties and display as VALVE. See section 2.8.3.13, VALVE

2.8.3.15 MOTOR

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 54		
Links/CollectValue	GroupDisplay_withASD .CollectValue	AdvancedStatusDis.	.EventState
Links/QMAN_AUT	Mode.Status1	AdvancedStatusDis.	Display: Auto/Manual
Links/LOCK	Interlock.Status1	AdvancedStatusDis.	Display: Lock
Links/QRUN	Motor_Status.Status1	AdvancedStatusDis.	Display: Motor
Links/QSTOP	Motor_Status.Status2	AdvancedStatusDis.	Display: Motor

Left-click opens the MOTOR faceplate, right-click calls the corresponding INTERLOK faceplate.

The name of the INTERLOK block is stored as script transfer parameter, see Chapter 2.4 "Scripts"

The default block name is "L“. The INTERLOK block and the MOTOR block must be placed in the same CFC chart.

2.8.3.16 MOT_SPED

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 53		
Links/CollectValue	GroupDisplay_withASD .CollectValue	AdvancedStatusDis.	.EventState
Links/QMAN_AUT	Mode.Status1	AdvancedStatusDis.	Display: Auto/Manual
Links/LOCK	Interlock.Status1	AdvancedStatusDis.	Display: Lock
Links/QRUN	Motor_Status.Status1	AdvancedStatusDis.	Display: Motor
Links/QSTOP	Motor_Status.Status2	AdvancedStatusDis.	Display: Motor
Links/QSPEED	Motor_Status.Status3	AdvancedStatusDis.	Display: Motor
Links/QSTOPING	Motor_Status.Status4	AdvancedStatusDis.	Display: Motor

A left-click calls the MOT_SPED faceplate, a right-click the corresponding INTERLOK faceplate.

The name of the INTERLOK block is stored as script transfer parameter, see Chapter 2.4 "Scripts"

The default block name is "L“. The INTERLOK block and MOT_SPED must be placed into the same CFC chart.

2.8.3.17 MOT_REV

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 53		
Links/CollectValue	GroupDisplay_withASD .CollectValue	AdvancedStatusDis.	.EventState
Links/QMAN_AUT	Mode.Status1	AdvancedStatusDis.	Display: Auto/Manual
Links/LOCK	Interlock.Status1	AdvancedStatusDis.	Display: Lock
Links/QRUN	Motor_Status1.Status1	AdvancedStatusDis.	Display: Motor
Links/QSTOP	Motor_Status1.Status2	AdvancedStatusDis.	Display: Motor
Links/QDIR	Motor_Status1.Status3	AdvancedStatusDis.	Display: Motor

A left-click calls the MOT_REV faceplate, a right-click the corresponding INTERLOK faceplate.

The name of the INTERLOK block is stored as script transfer parameter, see Chapter 2.4 "Scripts"

The default block name is "L“. The INTERLOK block and MOT_SPED must be placed into the same CFC chart.

2.8.3.18 INTERLOK

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 108 / Height = 20		
Links/Link	Lock.ActualStatus	AddDispl.	Lock icon

2.8.3.19 OP_D3

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 45		
Links/Output1	StatusDisplay1.Display	StatusDisplay.	.Q1
Links/Output2	StatusDisplay 2.Display	StatusDisplay.	.Q2
Links/Output3	StatusDisplay 3.Display	StatusDisplay.	.Q3

2.8.3.20 OP_D

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 45		
Links/Status	StatusDisplay1.ActualStatus	StatusDisplay	.Q0

2.8.3.21 OP_TRIG

See also section 2.8.3.1 "General Properties"

Properties	Element and property in the user object	Object	Description
Geometry	Width = 90 / Height = 40		
Links/Status	StatusDisplay1.ActualStatus	StatusDisplay	.SIGNAL

2.8.3.22 DIG_MON

See also section 2.8.3.1 "General Properties"

Properties	**Element and property in the user object**	Object	**Description**
Geometry	Width = 90 / Height = 45		
Links/Status	StatusDisplay1.ActualStatus	StatusDisplay	.Q
Links/CollectValue	GroupDisplay3.CollectValue	GroupDisplay	.EventState

3 Creating the Online Help

Requirements

You require the following:

- The ASCII "Notepad" editor integrated in WINDOWS or a similar editor for creating a registry file.
- A tool for authoring the help topics (for example, "RoboHelp").

3.1 Structure of the Help File

If you want to create an online help system for your blocks, you write a help file with the aid of a help authoring system. You can select any name for the file, however it is advisable to use the name of your library (or a common name that you have used for your blocks), for example "**MYLIB__b.HLP**".

Create a separate topic for each of your blocks. You must then define the entry address (Topic-ID + MAP #) for each topic in the online help and enter them in the registry file (see also Section 3.2). These must be unique within the online help system but can otherwise be used as required.

If you have a relatively large library, you can also create an hm file containing all the IDs used. The RoboHelp authoring tool can use this file when assigning MAP IDs.

Entries in the hm file:

```
// Header File for Online Help on Mylib Function Blocks
//
#define CONTROL         0x10                          // dec. 16
#define CONTROL2        0x11                          // dec. 17
#define CONTROL3        0x12                          // dec. 18
....
```

Apart from the help text, each help topic includes the following information:

Topic title	Title of the help topic for this block (normally the same as the block name)
Topic ID	Name and MAP ID of the topic (as specified in the registry file see Figure 1-3)
Index	Keywords with which the user can jump to a topic from the help index

The help system can consist of two files, an HLP file (help topics) and a CNT file (contents).

The CNT file is useful if the block help system is not intended solely as a context-sensitive help system (F1 with the block selected). If you have a library with several blocks, the individual topics can be listed in a table of contents. This allows the user to select the topics of other blocks without the block having to exist.

The CNT file can also be included in the CNT file of another help project using an INCLUDE statement (for example, ":include Mylib__b.cnt"). The included CNT file is then displayed in the contents of the other help project if both are installed in the same folder.

If you want to provide your online help system in several languages, you must create a separate help file for each language required. In PCS 7, the name consists of 8 characters with the last character being used as the language identifier:

a	German	
b	English	
c	French	
d	Spanish	Not currently supported by PCS 7
e	Italian	
y	Not language-dependent	For example, for the reg file

By registering (see also Section 3.2), PCS 7 then calls the language as selected in the "**Options > Customize > National Language**" dialog.

Finally, you copy the help file (and, if it exists, the CNT file) to the subfolder of the STEP 7 folder in which your library or the project with your blocks is installed.

3.2 Structure of the Registry File

Using the ASCII editor, write a registry file that will enter the information for your blocks in the WINDOWS registry. You can select any name for the registry file, however it is generally advisable to use the name of your library (or a common name that you have used for your blocks), for example "**Mylib_y.reg**".

Example of a reg file for 3 blocks and 5 language versions (Spanish and Italian with English help text).

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\mylib\ABC
]
"Version"="0.1"
"VersionDate"="03.05.2000"
"HelpFileGerman"="S7libs\\mylib\\MYLIB__a.hlp"
"HelpFileEnglish"="S7libs\\mylib\\MYLIB__b.hlp"
"HelpFileFrench"="S7libs\\mylib\\MYLIB__c.hlp"
"HelpFileSpanish"="S7libs\\mylib\\MYLIB__b.hlp"
"HelpFileItalian"="S7libs\\mylib\\MYLIB__b.hlp"

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\mylib\ABC
\XYZ]
"CONTROL"=dword:00000010
"CONTROL2"=dword:00000011
"CONTROL3"=dword:00000012
```

Note:

Please note that incorrect entries in the registry can lead to problems in program execution or can result in a function not being executed.
You should therefore use the key as shown in this example.

The following values must be entered in the registry key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\NameOfLibrary\Author]

Here, the library name stands for the name you selected for your library (here: mylib). This is the same as the name of the STEP 7 subfolder in which your help file is stored. You library is displayed in the CFC editor under this name. **Author** stands for the name you specified for the AUTHOR attribute in the block header (here: ABC).

Version
This contains the version number of the entire library. This entry is optional.

VersionDate
This contains the date on which the complete library was created. This entry is optional.

Path to the help file
This contains the path relative to the STEP 7 folder for the required help file, for example:

"HelpFileEnglish"="S7libs\\mylib\\MYLIB__b.hlp".

Please note that double delimiters must be specified (\\ ). Using this entry, the help file matching the language set in the SIMATIC Manager is called.

Note:

Italian and Spanish are not yet supported by PCS 7, so the help system is called in English language.

Then enter the following key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\Name OfLibrary \Author\Family]

Where **Family** stands for the name you specified for the FAMILY attribute in the block header (here: XYZ).

Below this, you must specify the block name, as specified in the header, for each block (see Figure 1-3), with the entry address in the Help file, for example: "CONTROL"=dword:00000010. The entry address is the number of the Topic ID "CONTROL"=dword:00000010.

If you have grouped your blocks in several families, you must insert a separate key in the registry file for each family.

Once the registry file has been executed (for example by double clicking it) when you next select a block in the CFC chart or in the SIMATIC Manager and then press the F1 key, the corresponding help file is located and displayed using the selected language and the block attributes AUTHOR, FAMILY, and FUNCTION_BLOCK in the WINDOWS registry.

3.3 Special Features for Creating Helps for SFC Types

Different from or as a supplement to the information in chapters 3.1 and 3.2, you have to take the following into consideration when creating online helps for SFC types:

Storage location

You have to copy the Help for the SFC types to your installation directory. We recommend you store it in the already existing "S7Hlp" folder. This is the folder,in which the SFC Helps are also contained. This is the only way to ensure that, you can also navigate to the SFC Help from your self-created Help, see below. The following example for the registration refers tho the location.

Entries for the key

Author	The entry for "Author" is assigned by SFC because it compiles the SFC types into executable blocks. Always: **ES_SFC**.
Family	For the family name, refer to the SFC type properties catalog from the "Family" field (no empty string). In the example: **SFC**
Name	For the SFC type name, refer to SFC type the properties catalog from the "Name" field (no empty string). In the example: "**Dosi1"** and **"Heat1"**

Example of a registry file

For 2 block types and 2 language versions.

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\SFCTypes]

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\SFCTypes\ES_SF
C]
"HelpFileGerman"="S7Hlp\\SFC_Typa.hlp"
"HelpFileENGLISH"="S7Hlp\\SFC_Typb.hlp"
"Version"="1.1"
"VersionDate"="14.11.2003"

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\SFCTypes\ES_SF
C\SFC]
"Dosi1"=dword:00000888
"Heat1"=dword:00000889
```

Navigating to the SFC Help

To be able to navigate to the SFC Help from your self-created Help, you can insert a jump to the (internal) table of contents of the SFC Help in the individual topics. This jump looks like this:

SFC Help!JumpID(`s7jsfcab.hlp',`IDH_CONTENTS')

Note: The jump address to the green double underlined text → !JumpID(`s7jsfcab.hlp',`IDH_CONTENTS') is formated as a hidden text.

4 Creating a Library and Setup

Requirements

To create a distributable library including the setup, you require a program for creating installation programs, for example "InstallShield".

4.1 Creating a Library

If you want to put your blocks and /or SFC types together in a library, follow the steps outlined below:

1. Create a new S7 library and create an S7 program in it.
2. Enter the names and numbers of your blocks as well as the corresponding comments in the icon table of the library (this does not apply to SFC types).
3. Source files:
   If you want to ship the source files of the blocks as well, copy the source files from the sources folder of your project to the sources folder of the library.
4. Blocks:
   Copy your blocks from the block folder of your project to the block folder of the library.

   Note:

   - If you call blocks in your multiple instance blocks that are not generally available (SFBs, SFCs), copy these to the block folder of the library as well.
   - You may not copy the blocks created during the SFC types compilation.
5. SFC types:
   Create a chart folder, if does not exist yet.
   Copy the SFC types from the chart folder of your project to the chart folder of the library.
6. Note: The associated blocks of the SFC types will also be copied and saved in the block folder.

4.2 Creating a Setup

If you want to install your library using setup on the target computer, write an installation script with the setup authoring tool that will perform the following actions:

- Copy the block library to the subfolder **S7LIBS** of the STEP 7 folder
- Copy the program **S7bin\S7ALIBXX.exe** in the STEP 7 folder to make the new library known to SIMATIC Manager
- Copy the help file (HLP and CNT file) to the subfolder of the STEP 7 folder into which the block library was copied (for example, the subfolder **S7LIBS\MYLIB**)
- Call the registry file belonging to the help file
- Copy the prototype pictures to in the **options\pdl**\FaceplateDesigner_V6 subfolder of the the WinCC directory.
- Copy the scripts to any subfolder in the **aplib** subfolder in the WinCC folder. It is advisable for this folder to have the same name as the folder into which the block library was copied (for example, **options\pdl\mylib**).
- Create an uninstall option

Note that the block library and the online help system can only be installed when STEP 7 exists on the target computer. The prototype pictures can only be installed in a subfolder of WinCC. Make sure that you query the existence of STEP 7 and WinCC in your installation dialog, for example by querying the following entries in the registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7
the name **STEP7_VERSION** must be set to the value of the desired version (for example "5.2")
and in the key **HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\WinCC\Setup**
the name **Version** must be set to the value (for example, V6.0 SP1).

A Samples: Source code of blocks MEAS_MON, MOTOR and VALVE

A.1 MEAS_MON

```
// @(#) $Header::   L:/PCS7/PCS7LibV51.SW/TechBlocks/VCS/Algorithm/meas_mon.csv$
//Copyright (C) Siemens AG 1995. All Rights Reserved. Confidential

// PCS 7 Library Vx.x
// Function: Meas.value monitoring block
// Label Version 3.0
// Macro Version :V0.92
FUNCTION_BLOCK "MEAS_MON"
TITLE = 'Meas.value monitoring block'
//
{
        S7_tasklist:=        'OB100';
        S7_alarm_ui:=       '1';
        S7_m_c:=            'true'
}
AUTHOR:              TECHN61
NAME:                MEAS_MON
VERSION:             '3.0'
FAMILY:              CONTROL
KNOW_HOW_PROTECT

VAR_INPUT
OOS  {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='In Service';
        S7_string_1:='Out of Service'} :    BOOL := 0; // 1= Out of Service
M_SUP_AH     {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='Suppress HH=No';
        S7_string_1:='Suppress HH=Yes'} :   BOOL := 0; // 1=Suppress HH Alarm
M_SUP_AL     {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='Suppress LL=No';
        S7_string_1:='Suppress LL=Yes'} :   BOOL := 0; // 1=Suppress LL Alarm
M_SUP_WH     {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='Suppress H=No';
        S7_string_1:='Suppress H=Yes'} :    BOOL := 0; // 1=Suppress H Alarm (Warning)
M_SUP_WL     {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='Suppress L=No';
        S7_string_1:='Suppress L=Yes'} :    BOOL := 0; // 1=Suppress L Alarm (Warning)
CSF  {S7_dynamic:='true'} :     BOOL := 0; // Control System Fault 1=External Error
MSG_LOCK     {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // Enable 1=Messages locked
MO_PVHR  {S7_visible:='false';
        S7_m_c:='true';
        S7_shortcut:='Bar UL';
        S7_unit:=''} :  REAL := 110;       // High Limit Bar Range
MO_PVLR  {S7_visible:='false';
        S7_m_c:='true';
        S7_shortcut:='Bar LL';
        S7_unit:=''} :  REAL := -10;       // Low Limit Bar Range
```

```
USTATUS  {S7_visible:='false'} :    WORD := 0; // User Status Bits
U    {S7_qc:='true';
       S7_dynamic:='true';
       S7_m_c:='true';
       S7_shortcut:='PV';
       S7_unit:=''} :  REAL := 0;   // Analog Input (Measured Value)
QC_U : BYTE := 16#80;               // Quality Code for Input U
U_AH     {S7_link:='false';
       S7_edit:='para';
       S7_m_c:='true';
       S7_shortcut:='HH alarm';
       S7_unit:=''} :  REAL := 100; // HH Alarm Limit
U_WH     {S7_link:='false';
       S7_edit:='para';
       S7_m_c:='true';
       S7_shortcut:='H alarm';
       S7_unit:=''} :  REAL := 95;  // H Alarm Limit (Warning)
U_WL     {S7_link:='false';
       S7_edit:='para';
       S7_m_c:='true';
       S7_shortcut:='L alarm';
       S7_unit:=''} :  REAL := -3;  // L Alarm Limit (Warning)
U_AL     {S7_link:='false';
       S7_edit:='para';
       S7_m_c:='true';
       S7_shortcut:='LL alarm';
       S7_unit:=''} :  REAL := -5;  // LL Alarm Limit
HYS  {S7_link:='false';
       S7_m_c:='true';
       S7_shortcut:='Hysteresis';
       S7_unit:=''} :  REAL := 5;   // Hysteresis of Analog Input
MSG_EVID     {S7_visible:='false';
       S7_link:='false';
       S7_param :='false';
       S7_server:='alarm_archiv';S7_a_type:='alarm_8p'} :  DWORD := 0; // Message ID
BA_EN    {S7_visible:='false';
       S7_m_c:='true'} :   BOOL := 0;        // Batch Enable
OCCUPIED     {S7_visible:='false';
       S7_m_c:='true'} :   BOOL := 0;        // Occupied by Batch
BA_ID    {S7_visible:='false';
       S7_m_c:='true'} :   DWORD := 0;       // Batch ID
BA_NA    {S7_visible:='false';
       S7_m_c:='true'} :   STRING[32] := ''; // Batch Name
STEP_NO  {S7_visible:='false';
       S7_m_c:='true'} :   DWORD := 0;       // Batch Step Number
RUNUPCYC     {S7_visible:='false';
       S7_link:='false'} :     INT := 3;     // Number of Run Up Cycles
END_VAR

VAR_IN_OUT
AUX_PR05     {S7_visible:='false'} :    ANY; // Auxiliary Value 5
AUX_PR06     {S7_visible:='false'} :    ANY; // Auxiliary Value 6
AUX_PR07     {S7_visible:='false'} :    ANY; // Auxiliary Value 7
AUX_PR08     {S7_visible:='false'} :    ANY; // Auxiliary Value 8
AUX_PR09     {S7_visible:='false'} :    ANY; // Auxiliary Value 9
AUX_PR10     {S7_visible:='false'} :    ANY; // Auxiliary Value 10
END_VAR

VAR_OUTPUT
QERR     {S7_visible:='false';
       S7_m_c:='true'} :   BOOL := 1; // 1=Error
QH_ALM   {S7_dynamic:='true'} :     BOOL := 0; // 1=HH-Alarm active
QL_ALM   {S7_dynamic:='true'} :     BOOL := 0; // 1=LL Alarm active
QH_WRN   {S7_dynamic:='true'} :     BOOL := 0; // 1=H Alarm active (Warning)
QL_WRN   {S7_dynamic:='true'} :     BOOL := 0; // 1=L Alarm active (Warning)
QMSG_ERR     {S7_visible:='false';
       S7_dynamic:='true'} :   BOOL := 0;     // 1=Message ERROR
QMSG_SUP     {S7_visible:='false';
       S7_dynamic:='true';
       S7_m_c:='true'} :   BOOL := 0;         // 1=Message Suppression Active
MSG_STAT     {S7_visible:='false';
       S7_dynamic:='true'} :   WORD := 0;     // Message: STATUS output
MSG_ACK  {S7_visible:='false';
       S7_dynamic:='true'} :   WORD := 0;     // Message: ACK_STATE output
VSTATUS  {S7_visible:='false';
       S7_m_c:='true'} :   DWORD := 0;        // Status word
END_VAR
```

```
CONST
C_OOS := 0;        // 1= Out of Service
C_M_SUP_AH := 0;   // 1=Suppress HH Alarm
C_M_SUP_AL := 0;   // 1=Suppress LL Alarm
C_M_SUP_WH := 0;   // 1=Suppress H Alarm (Warning)
C_M_SUP_WL := 0;   // 1=Suppress L Alarm (Warning)
C_CSF := 0;        // Control System Fault 1=External Error
C_MSG_LOCK := 0;   // Enable 1=Messages locked
C_MO_PVHR := 110; // High Limit Bar Range
C_MO_PVLR := -10; // Low Limit Bar Range
C_USTATUS := 0;    // User Status Bits
C_U := 0;          // Analog Input (Measured Value)
C_QC_U := 16#80;   // Quality Code for Input U
C_U_AH := 100;     // HH Alarm Limit
C_U_WH := 95;      // H Alarm Limit (Warning)
C_U_WL := -3;      // L Alarm Limit (Warning)
C_U_AL := -5;      // LL Alarm Limit
C_HYS := 5;        // Hysteresis of Analog Input
C_AUX_PR05 := 0;   // Auxiliary Value 5
C_AUX_PR06 := 0;   // Auxiliary Value 6
C_AUX_PR07 := 0;   // Auxiliary Value 7
C_AUX_PR08 := 0;   // Auxiliary Value 8
C_AUX_PR09 := 0;   // Auxiliary Value 9
C_AUX_PR10 := 0;   // Auxiliary Value 10
C_MSG_EVID := 0;   // Message ID
C_BA_EN := 0;      // Batch Enable
C_OCCUPIED := 0;   // Occupied by Batch
C_BA_ID := 0;      // Batch ID
C_BA_NA := '';     // Batch Name
C_STEP_NO := 0;    // Batch Step Number
C_RUNUPCYC := 3;   // Lag: Number of Run Up Cycles
C_QERR := 1;       // 1=Error
C_QH_ALM := 0;     // 1=HH-alarm Active
C_QL_ALM := 0;     // 1=LL Alarm Active
C_QH_WRN := 0;     // 1=H Alarm active (Warning)
C_QL_WRN := 0;     // 1=L Alarm Active (Warning)
C_QMSG_ERR := 0;   // 1=Message ERROR
C_QMSG_SUP := 0;   // 1=Message Suppression Active
C_MSG_STAT := 0;   // Message: STATUS Output
C_MSG_ACK := 0;    // Message: ACK_STATE output
C_VSTATUS := 0;    // Status word
END_CONST

// Static Variables
VAR
siRUNUPCNT:    int := 0;   // Counter for RUNUPCYC editing
 sb_SIG_1: bool    := FALSE;   //Merker ALARM_8P Signal 1
 sb_SIG_2: bool    := FALSE;   //Merker ALARM_8P Signal 2
 sb_SIG_3: bool    := FALSE;   //Merker ALARM_8P Signal 3
 sb_SIG_4: bool    := FALSE;   //Merker ALARM_8P Signal 4
 sb_SIG_5: bool    := FALSE;   //Merker ALARM_8P Signal 5
ALARM_8P_1:    ALARM_8P;   // Multiple instance ALARM_8P
siBA_ID:   dword := 0;     // Old value BA_ID
sbyBA_NA:  array[1..32] of byte := 32(0);
VSTATUS_LOC : DWORD :=16#0;// Local static variable, in which the output VSTATUS
                           // is copied.
STEP_NO_LOC : DWORD;       // Local variable, in which the input STEP_NO is
                           // saved.
PV_IN_LOC : REAL;          // Local variable, in which the input PV_IN is saved.
dwDUMMY:    DWORD := 0;    // Stand-by
BA_ID_LOC : DWORD;         // Local static variable, in which the input BA_ID
                           // is copied.
U_LOC : REAL;  // Local static variable, in which the input U is copied, because of
               // message in ALARM_8P

// Variables for the status word "VSTATUS" and "USTATUS"
VSTATUS_STR AT VSTATUS_LOC :  STRUCT
                                USTATUS : WORD;
                                VSTATUS_LOW_BIT_8 : BOOL;
                                VSTATUS_LOW_BIT_9 : BOOL;
                                VSTATUS_LOW_BIT_10 : BOOL;
                                VSTATUS_LOW_BIT_11 : BOOL;
                                VSTATUS_LOW_BIT_12 : BOOL;
                                VSTATUS_LOW_BIT_13 : BOOL;
                                VSTATUS_LOW_BIT_14 : BOOL;
                                VSTATUS_LOW_BIT_15 : BOOL;
                                VSTATUS_LOW_BIT_0 : BOOL;
                                VSTATUS_LOW_BIT_1 : BOOL;
                                VSTATUS_LOW_BIT_2 : BOOL;
```

```
                                VSTATUS_LOW_BIT_3 : BOOL;
                                VSTATUS_LOW_BIT_4 : BOOL;
                                VSTATUS_LOW_BIT_5 : BOOL;
                                VSTATUS_LOW_BIT_6 : BOOL;
                                VSTATUS_LOW_BIT_7 : BOOL;
                              END_STRUCT;
END_VAR

// Temporary Variables
VAR_TEMP
 pbALARM:  BOOL;       // Call up ALARM_8P
 pbM_SUP:  BOOL;       // Message suppression
 pb_SIG_1,pb_SIG_2,pb_SIG_3,pb_SIG_4:BOOL;
TOP_SI:    STRUCT
   EV_CLASS:   BYTE;
   EV_NUM:     BYTE;
   PRIORITY:   BYTE;
   NUM:        BYTE;
   TYP2_3:     BYTE;
   TYP1:       BYTE;
   ZI1:        WORD;
   ZI2_3:      DWORD;
   END_STRUCT;
START_UP_SI:   STRUCT
   EV_CLASS:   BYTE;
   EV_NUM:     BYTE;
   PRIORITY:   BYTE;
   NUM:        BYTE;
   TYP2_3:     BYTE;
   TYP1:       BYTE;
   ZI1:        WORD;
   ZI2_3:      DWORD;
   END_STRUCT;
 ERR :     INT;    // Error at startup
END_VAR

BEGIN
    // Write VSTATUS, STEP_NO resave as STEP_NO_LOC and
      // BA_ID resave as BA_ID_LOC.
    // Supply of VSTATUS output variables with the inputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_0 := OCCUPIED;
    VSTATUS_STR.VSTATUS_LOW_BIT_1 := BA_EN;
    VSTATUS_STR.VSTATUS_LOW_BIT_2 := MSG_LOCK;
    // Supply of VSTATUS output variables (HIGH BYTE) with the values,
      // which come from the user via the input variable USTATUS (from external).
    VSTATUS_STR.USTATUS := USTATUS;
    STEP_NO_LOC := STEP_NO; // Resave, due to output of STEP_NO_LOC in ALARM_8P
    BA_ID_LOC := BA_ID;     // Resave, due to output of BA_ID_LOC in ALARM_8P
    U_LOC := U;             // Resave, due to output of U_LOC in ALARM_8P
    ERR := RD_SINFO (TOP_SI := TOP_SI, START_UP_SI := START_UP_SI);
    // Read out start info
    pbM_SUP := MSG_LOCK;
    IF TOP_SI.NUM = 100   THEN     // When startup
        siRUNUPCNT := RUNUPCYC;    // Saving the value of the RUNUPCYC input
        // Initialization outputs
        QMSG_ERR := C_QMSG_ERR;
        QMSG_SUP := C_QMSG_SUP;
        MSG_STAT := C_MSG_STAT;
        MSG_ACK  := C_MSG_ACK;
        QH_WRN := C_QH_WRN;
        QL_WRN := C_QL_WRN;
        QH_ALM := C_QH_ALM;
        QL_ALM := C_QL_ALM;
        // pbALARM := NOT OOS; // Initialization first call ALARM_8P
        pb_SIG_1:= QH_ALM;     // Alarm high
        pb_SIG_2:= QH_WRN;     // Warning high
        pb_SIG_3:= QL_WRN;     // Warning low
        pb_SIG_4:= QL_ALM;     // Alarm low
        pbALARM :=TRUE;        // Initialization ALARM
    ELSE;
        // LIMITS_P.1 f¸r Alarmpr¸fung (optimiert)
        IF (U <= U_AL) THEN            // Low limit responded
            QL_ALM := TRUE;
        ELSE;
            IF (U >= (U_AL+HYS)) THEN  // Reset low limit responded
                QL_ALM := FALSE;
            ELSE;                      // QL_ALM remains unchanged
```

```
        END_IF;
    END_IF;
    IF (U >= U_AH) THEN           // High limit responded
                    QH_ALM := TRUE;
    ELSE;
        IF (U <= (U_AH-HYS)) THEN // Reset high limit responded
            QH_ALM := FALSE;
        ELSE;                     // QH_ALM remains unchanged
        END_IF;
    END_IF;
    // LIMITS_P.2 for Warning check (optimized)
    IF (U <= U_WL) THEN           // Low limit responded
        QL_WRN := TRUE;
    ELSE;
        IF (U >= (U_WL+HYS)) THEN // Reset low limit responded
            QL_WRN := FALSE;
        ELSE;                     // QL_WRN remains unchanged
        END_IF;
    END_IF;
    IF (U >= U_WH) THEN           // High limit responded
        QH_WRN := TRUE;
    ELSE;
        IF (U <= (U_WH-HYS)) THEN // Reset high limit responded
            QH_WRN := FALSE;
        ELSE;                     // QH_WRN remains unchanged
        END_IF;
    END_IF;
    IF siRUNUPCNT = 0  THEN       // Initialize alarms
        IF M_SUP_AH OR MSG_LOCK THEN
            pb_SIG_1:= 0;         // Report possible outgoing
        ELSE;
            pb_SIG_1:= QH_ALM;    // Alarm high
        END_IF;
        IF M_SUP_WH OR MSG_LOCK THEN
            pb_SIG_2:= 0;         // Report possible outgoing
        ELSE;
            pb_SIG_2:= QH_WRN;    // Warning high
        END_IF;
        IF M_SUP_WL OR MSG_LOCK THEN
            pb_SIG_3:= 0;         // Report possible outgoing
        ELSE;
            pb_SIG_3:= QL_WRN;    // Warning low
        END_IF;
        IF M_SUP_AL OR MSG_LOCK THEN
            pb_SIG_4:= 0;         // Report possible outgoing
        ELSE;
            pb_SIG_4:= QL_ALM;    // Alarm low
        END_IF;
        // pbALARM := (sb_SIG_1 <> pb_SIG_1) OR (sb_SIG_2 <>  pb_SIG_2)
          // OR (sb_SIG_3 <>  pb_SIG_3) OR (sb_SIG_4 <>  pb_SIG_4)
            // OR (sb_SIG_5 <>  CSF);
        pbALARM :=TRUE;           // Initialization ALARM
    ELSE;
        pbALARM :=FALSE;          // Initialization no ALARM
        pbM_SUP := TRUE;
        siRUNUPCNT := siRUNUPCNT - 1;
    END_IF;
END_IF;
IF pbALARM  THEN                  // Call up ALARM_8P
    IF siBA_ID <> BA_ID_LOC THEN
        // STRING variables may not be interconnected to ALARM8_P as auxiliary
          //process values, therefore transferred in ARRAY OF BYTE.
        FOR ERR := 1 TO 32
         DO
            sbyBA_NA[ERR] := 0;   // Delete array as default
        END_FOR;
        ERR := BLKMOV (SRCBLK:= BA_NA,DSTBLK:=sbyBA_NA);
        siBA_ID := BA_ID_LOC;     // Save modified BA_ID
    END_IF;
    // Call ALARM_8P with lock logic (MSG_LOCK).
    ALARM_8P_1(    EN_R := TRUE,  // Update the ACKL_STATE output
                   ID := 16#EEEE, // PMC communication channel
       EV_ID:= MSG_EVID,
       SIG_1:= pb_SIG_1,
       SIG_2:= pb_SIG_2,
       SIG_3:= pb_SIG_3,
       SIG_4:= pb_SIG_4,
       SIG_5:= CSF,
       SIG_6:= 0,
```

```
            SIG_7:= 0,
            SIG_8:= 0,
            SD_1 := sbyBA_NA,
            SD_2 := STEP_NO_LOC,
            SD_3 := BA_ID_LOC,
            SD_4 := U_LOC,
            SD_5 := AUX_PR05,
            SD_6 := AUX_PR06,
            SD_7 := AUX_PR07,
            SD_8 := AUX_PR08,
            SD_9 := AUX_PR09,
            SD_10 := AUX_PR10);
        QMSG_ERR := ALARM_8P_1.ERROR;
        MSG_STAT := ALARM_8P_1.STATUS;
        MSG_ACK := ALARM_8P_1.ACK_STATE;
        IF (NOT QMSG_ERR) THEN          // Note historical signals.
            sb_SIG_1:= pb_SIG_1;
            sb_SIG_2:= pb_SIG_2;
            sb_SIG_3:= pb_SIG_3;
            sb_SIG_4:= pb_SIG_4;
            sb_SIG_5:= CSF;
        END_IF;
    END_IF;
    IF (MSG_STAT = 21) THEN             // Block locked
        pbM_SUP := TRUE;
    END_IF;
    QMSG_SUP := pbM_SUP;
    QERR := NOT OK;                     // Note negated OK-Flag result in the block.
    // Power supply of the VSTATUS output variable with the outputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_14 := QMSG_SUP OR M_SUP_AH OR M_SUP_WH OR M_SUP_WL OR
M_SUP_AL ;
    VSTATUS_STR.VSTATUS_LOW_BIT_15 := OOS;
    VSTATUS := VSTATUS_LOC;
END_FUNCTION_BLOCK
```

A.2 MOTOR

```
// @(#) $Header::   L:/PCS7/PCS7LibV51.SW/TechBlocks/VCS/Algorithm/motor.csv   $
//Copyright (C) Siemens AG 1995-1997. All Rights Reserved. Confidential

PCS 7 Library Vx.x
// Function: motor
// Label Version 3.0
// Macro Version :V0.92

FUNCTION_BLOCK "MOTOR"
TITLE = 'motor '
//
{
        S7_tasklist:=        'OB100';
        S7_alarm_ui:=       '1';
        S7_m_c:=            'true'
}
AUTHOR:              TECHN61
NAME:                MOTOR
VERSION:             '4.0'
FAMILY:              CONTROL
KNOW_HOW_PROTECT

VAR_INPUT
OOS  {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='In Service';
        S7_string_1:='Out of Service'} :    BOOL := 0; // 1= Out of Service
LOCK     {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // 1=Lock to OFF
LOCK_ON  {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // 1=Lock to ON
AUTO_ON  {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // AUTO Mode:1=ON, 0=OFF
L_RESET  {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // Linkable Input RESET
MSS  {S7_dynamic:='true'} :     BOOL := 1; // Motor Protecting Switch: 0=Active
CSF  {S7_dynamic:='true'} :     BOOL := 0; // Control System Fault 1=External Error
FB_ON    {S7_qc:='true';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0; // Feedback: 1=ON
QC_FB_ON : BYTE := 16#80;              // Quality Code for FB_ON
QC_QSTART_I : BYTE := 16#80;           // Quality Code for Input QSTART
ON_OP_EN   {S7_visible:='false'} :   BOOL := 1; // Enable 1=Operator may input "ON"
OFFOP_EN   {S7_visible:='false'} :   BOOL := 1; // Enable 1=Operator for "OFF"
MANOP_EN   {S7_visible:='false'} :   BOOL := 1; // Enable: 1=Operator may input "MANUAL"
AUTOP_EN   {S7_visible:='false'} :   BOOL := 1; // Enable: 1=Operator may input "AUTO"
LIOP_SEL   {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // Select: 1=Linking, 0=Operator Active
AUT_L    {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // Linkable Input for  MANUAL/AUTO Mode
MONITOR  {S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='Monitoring=Off';
        S7_string_1:='Monitoring=On'} :     BOOL := 1;  // Select: 1=Monitoring ON,
                                                        // 0=Monitoring OFF
TIME_MON     {S7_link:='false';
        S7_edit:='para';
        S7_m_c:='true';
        S7_shortcut:='Mon. Time';
        S7_unit:='s'} :    REAL := 3;          // Monitoring Time for ON [s]
SAMPLE_T     {S7_visible:='false';
        S7_sampletime:='true'} :    REAL := 1; // Sample Time [s]
MSG_EVID     {S7_visible:='false';
        S7_link:='false';
        S7_param :='false';
        S7_server:='alarm_archiv';S7_a_type:='alarm_8p'} :  DWORD := 0; // Message ID
BA_EN    {S7_visible:='false';
        S7_m_c:='true'} :   BOOL := 0;        // Batch Enable
OCCUPIED     {S7_visible:='false';
        S7_m_c:='true'} :   BOOL := 0;        // Occupied by Batch
```

```
BA_ID    {S7_visible:='false';
         S7_m_c:='true'} :     DWORD := 0;      // Batch ID
BA_NA    {S7_visible:='false';
         S7_m_c:='true'} :   STRING[32] := ''; // Batch Name
STEP_NO  {S7_visible:='false';
         S7_m_c:='true'} :   DWORD := 0;       // Batch Step Number
RUNUPCYC     {S7_visible:='false';
         S7_link:='false'} :     INT := 3; // Lag: Number of Run Up Cycles
START_OFF    {S7_visible:='false'} :    BOOL := 1; // 1=Start up with Motor OFF
FAULT_OFF    {S7_visible:='false'} :    BOOL := 1; // 1=In case of Fault: Motor OFF
MSS_OFF  {S7_visible:='false'} :    BOOL := 1; // 1=In case of MSS-Fault: Motor OFF
USTATUS  {S7_visible:='false'} :    WORD := 0; // User STATUS Bits
END_VAR

VAR_IN_OUT
RESET    {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='0';
         S7_string_1:='Error=Reset'} :   BOOL := 0; // Operator Input Error Reset
MAN_ON   {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='Motor=Stop';
         S7_string_1:='Motor=Start'} :   BOOL := 0; // Operator Input: 1=ON, 0=OFF
AUT_ON_OP    {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='Mode=Manual';
         S7_string_1:='Mode=Auto'} :     BOOL := 0; // Operator Input Mode 1=AUTO,
                                                    // 0= MANUAL
AUX_PR04     {S7_visible:='false'} :    ANY; // Auxiliary Value 4
AUX_PR05     {S7_visible:='false'} :    ANY; // Auxiliary Value 5
AUX_PR06     {S7_visible:='false'} :    ANY; // Auxiliary Value 6
AUX_PR07     {S7_visible:='false'} :    ANY; // Auxiliary Value 7
AUX_PR08     {S7_visible:='false'} :    ANY; // Auxiliary Value 8
AUX_PR09     {S7_visible:='false'} :    ANY; // Auxiliary Value 9
AUX_PR10     {S7_visible:='false'} :    ANY; // Auxiliary Value 10
END_VAR

VAR_OUTPUT
QERR     {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 1; // 1=Error
QMSS_ST  {S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Unacknowledged Motor Protective Switch
QMON_ERR     {S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // 1=Monitoring Error
QGR_ERR  {S7_dynamic:='true';
         S7_contact:='true'} :   BOOL := 0; // 1=Group Error
QOP_ERR  {S7_visible:='false';
         S7_dynamic:='true'} :   BOOL := 0; // 1=Operator Error
QRUN     {S7_dynamic:='true';
         S7_contact:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Motor running
QSTOP    {S7_dynamic:='true';
         S7_contact:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Motor STOP
QSTART   {S7_qc:='true';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Control Output 1=START Active
QC_QSTART : BYTE := 16#80; // Quality Code for Output QSTART
QON_OP   {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Operator enabled for "ON"
QOFF_OP  {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Operator enabled for "OFF"
QMAN_AUT     {S7_dynamic:='true';
         S7_contact:='true';
         S7_m_c:='true'} :   BOOL := 0; // 1=AUTO, 0=MANUAL Mode
QMANOP   {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Oper. ena. for "MANUAL" Mode
```

```
QAUTOP   {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0; // Status: 1=Operator enabled for "AUTO"
QMSG_ERR     {S7_visible:='false';
         S7_dynamic:='true'} :   BOOL := 0; // 1=Message ERROR
QMSG_SUP     {S7_visible:='false';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0;     // 1=Message Suppression Active
MSG_STAT     {S7_visible:='false';
         S7_dynamic:='true'} :   WORD := 0; // Message: STATUS Output
MSG_ACK  {S7_visible:='false';
         S7_dynamic:='true'} :   WORD := 0; // Message: ACK_STATE output
VSTATUS  {S7_visible:='false';
         S7_m_c:='true'} :   DWORD := 0;    // Status word
END_VAR

CONST
C_OOS := 0;              // 1= Out of Service
C_LOCK := 0;             // 1=Lock to OFF
C_LOCK_ON := 0;          // 1=Lock to ON
C_AUTO_ON := 0;          // AUTO Mode:1=ON, 0=Off
C_RESET := 0;            // Operator Input Error Reset
C_L_RESET := 0;          // Linkable Input RESET
C_MSS := 1;              // Motor Protecting Switch: 0=Active
C_CSF := 0;              // Control System Fault 1=External Error
C_FB_ON := 0;            // Feedback: 1=ON
C_QC_FB_ON := 16#80;     // Quality Code for FB_ON
C_QC_QSTART_I := 16#80; // Quality Code for Input QSTART
C_MAN_ON := 0;           // Operator Input: 1=ON, 0=OFF
C_ON_OP_EN := 1;         // Enable 1=Operator may input ON
C_OFFOP_EN := 1;         // Enable 1=Operator for "OFF"
C_AUT_ON_OP := 0;        // Operator Input Mode 1=AUTO, 0= MANUAL
C_MANOP_EN := 1;         // Enable: 1=Operator may input MANUAL
C_AUTOP_EN := 1;         // Enable: 1=Operator may input AUTO
C_LIOP_SEL := 0;         // Select: 1=Linking, 0=Operator Active
C_AUT_L := 0;            // Linkable Input for  MANUAL/AUTO Mode
C_MONITOR := 1;          // Select: 1=Monitoring ON, 0=Monitoring OFF
C_TIME_MON := 3;         // Monitoring Time for ON [s]
C_SAMPLE_T := 1;         // Sample Time [s]
C_AUX_PR04 := 0;         // Auxiliary Value 4
C_AUX_PR05 := 0;         // Auxiliary Value 5
C_AUX_PR06 := 0;         // Auxiliary Value 6
C_AUX_PR07 := 0;         // Auxiliary Value 7
C_AUX_PR08 := 0;         // Auxiliary Value 8
C_AUX_PR09 := 0;         // Auxiliary Value 9
C_AUX_PR10 := 0;         // Auxiliary Value 10
C_MSG_EVID := 0;         // Message ID
C_BA_EN := 0;            // Batch Enable
C_OCCUPIED := 0;         // Occupied by Batch
C_BA_ID := 0;            // Batch ID
C_BA_NA := '';           // Batch Name
C_STEP_NO := 0;          // Batch Step Number
C_RUNUPCYC := 3;         // Lag: Number of Run Up Cycles
C_START_OFF := 1;        // 1=Start up with Motor OFF
C_FAULT_OFF := 1;        // 1=In case of fault: Motor OFF
C_MSS_OFF := 1;          // 1=In case of MSS fault: Motor OFF
C_USTATUS := 0;          // User STATUS Bits
C_QERR := 1;             // 1=Error
C_QMSS_ST := 0;          // Unacknowledged Motor Protective Switch
C_QMON_ERR := 0;         // 1=Monitoring Error
C_QGR_ERR := 0;          // 1=Group Error
C_QOP_ERR := 0;          // 1=Operator Error
C_QRUN := 0;             // Status: 1=Motor running
C_QSTOP := 0;            // Status: 1=Motor STOP
C_QSTART := 0;           // Control Output 1=START Active
C_QC_QSTART := 16#80;    // Quality Code for Output QSTART
C_QON_OP := 0;           // Status: 1=Operator enabled for "ON"
C_QOFF_OP := 0;          // Status: 1=Operator enabled for "OFF"
C_QMAN_AUT := 0;         // 1=AUTO, 0=MANUAL Mode
C_QMANOP := 0;           // Status: 1=Oper. enabled for "MANUAL" Mode
C_QAUTOP := 0;           // Status: 1=Operator enabled for "AUTO" Mode
C_QMSG_ERR := 0;         // 1=Message ERROR
C_QMSG_SUP := 0;         // 1=Message Suppression Active
C_MSG_STAT := 0;         // Message: STATUS Output
C_MSG_ACK := 0;          // Message: ACK_STATE output
C_VSTATUS := 0;          // Status word
END_CONST
```

```
// Static Variables
VAR
sbI_OD1:        BOOL := 0;       // Flag of old operating value for OP_D.1
sbI_OD2:        BOOL := 0;       // Flag of old operating value for OP_D.2
sbQ_OD1:        BOOL := 0;       // Binary output for OP_D.1
sbALT_LINK_I_OT1:  BOOL := 0;  // Old value of interconnectable input for OP_TRIG.1
sbALT_QSTART:       BOOL := 0;  // Historical process data for QSTART
 sb_SIG_1: BOOL     := FALSE;   // ALARM_8P signal 1 flag
 sb_SIG_2: BOOL     := FALSE;   // ALARM_8P signal 2 flag
 sb_SIG_3: BOOL     := FALSE;   // ALARM_8P signal 3 flag
srAktZeit:      REAL := 0;       // Time passed
siRUNUPCNT:         INT  := 0;  // Counter for RUNUPCYC editing

//------------------------ ALARM_8P.1 -----------------------------
ALARM_8P_1:         ALARM_8P;   // Multiple instanced ALARM_8P
dwDUMMY:    DWORD := 0;          // Stand-by
siBA_ID:   DWORD := 0;           // Old value BA_ID
sbyBA_NA:  ARRAY[1..32] OF BYTE := 32(0);

VSTATUS_LOC : DWORD :=16#0;  // Local variable, into which the VSTATUS output is copied
STEP_NO_LOC : DWORD;         // Local variable, into which the STEP_NO input is copied
BA_ID_LOC : DWORD;           // Local variable, into which the BA_ID input is copied

// Variables for STATUSWORT "VSTATUS" and "USTATUS"
VSTATUS_STR AT VSTATUS_LOC :  STRUCT
                                USTATUS : WORD;
                                VSTATUS_LOW_BIT_8 : BOOL;
                                VSTATUS_LOW_BIT_9 : BOOL;
                                VSTATUS_LOW_BIT_10 : BOOL;
                                VSTATUS_LOW_BIT_11 : BOOL;
                                VSTATUS_LOW_BIT_12 : BOOL;
                                VSTATUS_LOW_BIT_13 : BOOL;
                                VSTATUS_LOW_BIT_14 : BOOL;
                                VSTATUS_LOW_BIT_15 : BOOL;
                                VSTATUS_LOW_BIT_0 : BOOL;
                                VSTATUS_LOW_BIT_1 : BOOL;
                                VSTATUS_LOW_BIT_2 : BOOL;
                                VSTATUS_LOW_BIT_3 : BOOL;
                                VSTATUS_LOW_BIT_4 : BOOL;
                                VSTATUS_LOW_BIT_5 : BOOL;
                                VSTATUS_LOW_BIT_6 : BOOL;
                                VSTATUS_LOW_BIT_7 : BOOL;
                               END_STRUCT;
END_VAR

// Temporary variables
VAR_TEMP
TOP_SI:    STRUCT
   EV_CLASS:    BYTE;
   EV_NUM:      BYTE;
   PRIORITY:    BYTE;
   NUM:         BYTE;
   TYP2_3:      BYTE;
   TYP1:        BYTE;
   ZI1:         WORD;
   ZI2_3:       DWORD;
   END_STRUCT;
START_UP_SI:    STRUCT
   EV_CLASS:    BYTE;
   EV_NUM:      BYTE;
   PRIORITY:    BYTE;
   NUM:         BYTE;
   TYP2_3:      BYTE;
   TYP1:        BYTE;
   ZI1:         WORD;
   ZI2_3:       DWORD;
   END_STRUCT;
ERR :              INT;             // Startup error
pbLINK_ON_OD1:     BOOL;            // Operating/Interconnection flag for OP_D.1.
pbLINK_I_OD1:      BOOL;            // Interconnectable input for OP_D.1.
pbQOP_ERR:         BOOL;            // Operating error pointer for OP_D.1.
pbLINK_I_OT1:      BOOL;            // Interconnectable input for OP_TRIG.1.
pbQ_OT1:           BOOL;            // Binary output for OP_TRIG.1.
pbALARM:           BOOL;            // Call up ALARM_8P
pbM_SUP:           BOOL;            // Message suppression
END_VAR
```

```
BEGIN
    // Supply of the VSTATUS, resave STEP_NO as STEP_NO_LOC,
      // Resave BA_ID as BA_ID_LOC
    // Supply of the VSTATUS output variable with the inputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_0 := OCCUPIED;
    VSTATUS_STR.VSTATUS_LOW_BIT_1 := BA_EN;
    // Supply of the VSTATUS output variable (HIGH BYTE) with the values
      // that come from the user via the input variable USTATUS (from external).
    VSTATUS_STR.USTATUS := USTATUS;
    // Resave STEP_NO as STEP_NO_LOC and resave BA_ID as BA_ID_LOC
    STEP_NO_LOC := STEP_NO; // Resave, due to output of STEP_NO_LOC in ALARM_8P
    BA_ID_LOC := BA_ID;     // Resave, due to output of BA_ID_LOC in ALARM_8P
    ERR := RD_SINFO (TOP_SI := TOP_SI, START_UP_SI := START_UP_SI); // Read out
                                                                    // start info
    // pbM_SUP := FALSE;
    IF (TOP_SI.NUM = 100) AND START_OFF THEN
        siRUNUPCNT := RUNUPCYC;     // Save the value of the RUNUPCYC input
        // Initial states
        // Outputs to be set
        RESET     := C_RESET;
        MAN_ON    := C_MAN_ON;
        AUT_ON_OP := C_AUT_ON_OP;
        QERR      := C_QERR;
        QMSS_ST   := C_QMSS_ST;
        QMON_ERR  := C_QMON_ERR;
        QGR_ERR   := C_QGR_ERR;
        QOP_ERR   := C_QOP_ERR;
        QRUN      := C_QRUN;
        QSTOP     := C_QSTOP;
        QSTART    := C_QSTART;
        QC_QSTART := QC_QSTART_I; // Quality output described with value of input
        QON_OP    := C_QON_OP;
        QOFF_OP   := C_QOFF_OP;
        QMAN_AUT  := C_QMAN_AUT;
        QMANOP    := C_QMANOP;
        QAUTOP    := C_QAUTOP;
        QMSG_ERR  := C_QMSG_ERR;
                                  // QMSG_SUP := C_QMSG_SUP;
        MSG_STAT  := C_MSG_STAT;
        MSG_ACK   := C_MSG_ACK;
        // Static variables to be set
        sbI_OD1 := C_MAN_ON;         // Reset flag of old operating value for OP_D.1.
                                       // to zero
        sbI_OD2 := C_AUT_ON_OP;      // Reset flag of old operating value for OP_D.2.
                                       // to zero
        sbQ_OD1 := 0;                // Has to be set for incorrect operator
        sbALT_LINK_I_OT1 := L_RESET; // Previous value set by LINK_I
        sbALT_QSTART := C_QSTART;    // Set previous STOP-Modi
        srAktZeit := 0;              // Set acttime to NULL (zero)
        pbALARM :=TRUE;              // Initialization ALARM_8P
    ELSE;
        // MOTOR Algorithm
        // OP_D.2        Select manual or automatic mode
        pbQOP_ERR := FALSE; // Is only set to 1 in fault scenarios
        IF LIOP_SEL THEN
            // Interconnection
            IF AUT_ON_OP = sbI_OD2 THEN
                ;// No  operation occurs
            ELSE;
                pbQOP_ERR := TRUE;  // Prohibited operator
            END_IF;
            AUT_ON_OP := AUT_L;
            // Write AUT_L back to operator input
            QMAN_AUT := AUT_L;  // Interconnection
            QMANOP := FALSE;    // No operator allowed
            QAUTOP := FALSE;
        ELSE;
            // Operator
            IF (sbI_OD2 <> AUT_ON_OP) AND
               NOT ((AUT_ON_OP AND AUTOP_EN) OR (NOT(AUT_ON_OP) AND MANOP_EN)) THEN
                // Prohibited operator
                pbQOP_ERR := TRUE;
                AUT_ON_OP := sbI_OD2;
            ELSE;               // No or allowed operator
                QMAN_AUT := AUT_ON_OP;
            END_IF;
            QMANOP := MANOP_EN;
```

```
        // Copy the inputs to the outputs
        QAUTOP := AUTOP_EN;
    END_IF;
    sbI_OD2 := AUT_ON_OP;               // Note old operating values
    // OP_TRIG.1.
    pbLINK_I_OT1 := L_RESET AND MSS;    // Link_I for OP_TRIG.1.( RESET )
    // Activating RESET
    pbQ_OT1 := FALSE;           // Reset Q
    IF ( pbLINK_I_OT1 AND ( NOT sbALT_LINK_I_OT1 ) )   THEN
        pbQ_OT1 := TRUE;        // Set pbQ_OT1
    ELSE;
        IF RESET THEN
            pbQ_OT1 := TRUE;    // Set pbQ_OT1
        END_IF;
    END_IF;
    sbALT_LINK_I_OT1 := pbLINK_I_OT1;   // Save previous values
    RESET := FALSE;
    // Motor protector circuit-breaker
    // Establish motor protector circuit-breaker
    QMSS_ST := QMSS_ST AND (NOT pbQ_OT1) OR (NOT MSS);
    // Watchdog   - 1 -
    IF pbQ_OT1  THEN        // If RESET came in this cycle,
                              // The monitor error has to be corrected
        QMON_ERR := 0;
    END_IF;
    // Inputs for OP_D.1.(manual mode)
    pbLINK_ON_OD1 := LOCK OR LOCK_ON OR QMAN_AUT OR (QMSS_ST AND MSS_OFF)
                     OR (QMON_ERR AND FAULT_OFF);
    pbLINK_I_OD1 :=  (LOCK_ON OR (AUTO_ON AND (NOT(QMON_ERR AND FAULT_OFF))))
                     AND (NOT LOCK) AND (NOT(QMSS_ST AND MSS_OFF));
    // OP_D.1.
    // Outputs of the manual mode for control
    IF pbLINK_ON_OD1 THEN
        // Interconnection
        IF MAN_ON = sbI_OD1 THEN
            ; // No  operation occurred
        ELSE;
            pbQOP_ERR := TRUE;
            // Prohibited operator
        END_IF;
        MAN_ON := pbLINK_I_OD1;  // Write pbLINK_I_OD1 back to operator input
        sbQ_OD1 := pbLINK_I_OD1; // Interconnection
        QON_OP := FALSE;          // No operator allowed
        QOFF_OP := FALSE;
    ELSE;   // Operator
        IF (sbI_OD1 <> MAN_ON) AND
           NOT ((MAN_ON AND ON_OP_EN) OR (NOT(MAN_ON) AND OFFOP_EN)) THEN
            // Prohibited operator
            pbQOP_ERR := TRUE;
            MAN_ON := sbI_OD1;
        ELSE;   // No or allowed operator
                sbQ_OD1 := MAN_ON;
        END_IF;
        QON_OP := ON_OP_EN;     // Copy the inputs to the outputs
        QOFF_OP := OFFOP_EN;
    END_IF;
    sbI_OD1 := MAN_ON;          // Note old operating values
    QOP_ERR := pbQOP_ERR;       // Operator error
    // Watchdog   - 2 -
    // Switch the motor on or off within a configured timespan.
    IF (( sbALT_QSTART <> QSTART ) AND ( NOT QMON_ERR ))
       // Change through control
    THEN
        IF TIME_MON < SAMPLE_T   THEN
            srAktZeit := SAMPLE_T;
        ELSE;
            srAktZeit := TIME_MON;
            // Set acttime
        END_IF;
    END_IF;
    IF srAktZeit >= SAMPLE_T THEN
        srAktZeit := srAktZeit - SAMPLE_T; // Bring down time
    END_IF;
```

```
      IF MONITOR THEN
         // If the monitor is active, the change from Q_START has to be written
           // immediately to the output.
          IF FB_ON = QSTART THEN          // Before the monitor time expires,
                                            // same Feedback and QSTART
              srAktZeit := 0;
              IF QSTART THEN       // Report the corresponding feedback status
                  QRUN := 1;
              ELSE;
                  QSTOP := 1;
              END_IF;
          END_IF;
      END_IF;
      IF srAktZeit < SAMPLE_T  THEN // If the monitor time expires, does the  <=
          IF MONITOR  THEN          // watchdog then < instead function?
              IF ( FB_ON <> QSTART ) AND ( NOT pbQ_OT1 ) THEN
                 // After the monitor time, feedback and QSTART are different
                  QMON_ERR := 1;
              ELSE; END_IF;
          ELSE;
              IF QSTART THEN  // Set QRUN and QSTOP
                  QRUN := 1;
                  QSTOP := 0;
              ELSE;
                  QRUN := 0;
                  QSTOP := 1;
              END_IF;
          END_IF;
      END_IF;
      // Control signal
      sbALT_QSTART := QSTART;
      QGR_ERR := QMON_ERR OR QMSS_ST OR CSF;
      QSTART := sbQ_OD1 AND ((NOT(QMON_ERR AND FAULT_OFF)) OR LOCK_ON);
      QC_QSTART := QC_QSTART_I; // Describe quality output with value of input.
      // Watchdog   - 3 -
      // Set QRUN and QSTOP
      IF QSTART THEN  // Adapt QRUN and QSTOP to the previous value of QSTART.
          QSTOP := 0;
      ELSE;
          QRUN := 0;
      END_IF;
      IF siRUNUPCNT = 0 THEN
          // Initialize alarm
          // pbALARM := (sb_SIG_1 <> QMSS_ST)
          // OR (sb_SIG_2 <>  QMON_ERR) OR (sb_SIG_3 <>  CSF);
          pbALARM :=TRUE;   // Initialization ALARM_8P
      ELSE;
          siRUNUPCNT := siRUNUPCNT - 1;
          QMSG_SUP := TRUE;
          pbALARM := FALSE; // Initialization no ALARM
      END_IF;
  END_IF;
  IF pbALARM THEN           // Call up ALARM_8P
      IF siBA_ID <> BA_ID_LOC THEN
          // STRING variables may not be interconnected to ALARM8_P as auxiliary
            // process values, therefore transferred in ARRAY OF BYTE.
          FOR ERR := 1 TO 32
           DO
              sbyBA_NA[ERR] := 0;    // Delete array as default
          END_FOR;
          ERR := BLKMOV (SRCBLK:= BA_NA,DSTBLK:=sbyBA_NA);
          siBA_ID := BA_ID_LOC;      // Save modified BA_ID
      END_IF;
```

```
        // Call up ALARM_8P with lock logic (MSG_LOCK)
        ALARM_8P_1(    EN_R := TRUE,   // Update the ACKL_STATE output
                       ID := 16#EEEE,  // PMC communication channel
           EV_ID:= MSG_EVID,
           SIG_1:= QMSS_ST,
           SIG_2:= QMON_ERR,
           SIG_3:= CSF,
           SIG_4:= 0,
           SIG_5:= 0,
           SIG_6:= 0,
           SIG_7:= 0,
           SIG_8:= 0,
           SD_1 := sbyBA_NA,
           SD_2 := STEP_NO_LOC,
           SD_3 := BA_ID_LOC,
           SD_4 := AUX_PR04,
           SD_5 := AUX_PR05,
           SD_6 := AUX_PR06,
           SD_7 := AUX_PR07,
           SD_8 := AUX_PR08,
           SD_9 := AUX_PR09,
           SD_10 := AUX_PR10);
        QMSG_ERR := ALARM_8P_1.ERROR;
        MSG_STAT := ALARM_8P_1.STATUS;
        MSG_ACK := ALARM_8P_1.ACK_STATE;
        sb_SIG_1:= QMSS_ST;           // Note historical signals.
        sb_SIG_2:= QMON_ERR;
        sb_SIG_3:= CSF;
        QMSG_SUP := (MSG_STAT = 21);
    END_IF;
    QERR := NOT OK;
    // Power supply of the VSTATUS output variable with the outputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_3 := QMAN_AUT;
    VSTATUS_STR.VSTATUS_LOW_BIT_5 := QMSS_ST;
    VSTATUS_STR.VSTATUS_LOW_BIT_7 := QMON_ERR;
    VSTATUS_STR.VSTATUS_LOW_BIT_8 := LOCK;
    VSTATUS_STR.VSTATUS_LOW_BIT_9 := QRUN;
    VSTATUS_STR.VSTATUS_LOW_BIT_10 := QSTOP;
    VSTATUS_STR.VSTATUS_LOW_BIT_14 := QMSG_SUP;
    VSTATUS_STR.VSTATUS_LOW_BIT_15 := OOS;
    VSTATUS := VSTATUS_LOC;
END_FUNCTION_BLOCK
```

A.3 Valve

```
// @(#) $Header::   L:/PCS7/PCS7LibV51.SW/TechBlocks/VCS/Algorithm/valv$
//Copyright (C) Siemens AG 1995. All Rights Reserved. Confidential

(*------------------------------------------------------------------------------

Order of the part blocks:

1. OP_D.2
2. OP_TRIG.1
3. Watchdog RESET Logic
4. SEL_BOOL.1
5. OP_D.1
6. Watchdog (remaining)
7. XOR
8. MESSAGE

------------------------------------------------------------------------------*)

//PCS 7 Library Vx.x
//Funktion: Single-Drive/Dual-Feedback Valve
//Etikett-Version 3.0
//Makro-Version :V0.92

FUNCTION_BLOCK "VALVE"
TITLE = 'Single-Drive/Dual-Feedback Valve'
//
{
        S7_tasklist:=        'OB100';
        S7_alarm_ui:=       '1';
        S7_m_c:=            'true'
}
AUTHOR:              TECHN61
NAME:                VALVE
VERSION:             '3.0'
FAMILY:              CONTROL
KNOW_HOW_PROTECT

VAR_INPUT
OOS  {S7_visible:='false';
        S7_link:='false';
        S7_m_c:='true';
        S7_string_0:='In Service';
        S7_string_1:='Out of Service'} :    BOOL := 0; // 1= Out of Service
V_LOCK   {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // 1=Lock to SAVE position
VL_OPEN  {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // 1=Lock to OPEN
VL_CLOSE     {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // 1=Lock to CLOSE
AUTO_OC   {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // AUTO Mode:1=Open, 0=Close
SS_POS   {S7_dynamic:='true';
        S7_edit:='para';
        S7_m_c:='true'} :   BOOL := 0;     // Safe Position. 1=Open, 0=Close
START_SS     {S7_visible:='false'} :    BOOL := 1; // 1=Start with Safe State Position
                                                   // and Manual Mode.
FAULT_SS     {S7_visible:='false'} :    BOOL := 1; // 1=In Case of fault: Safe State
                                                   // Position.
L_RESET  {S7_dynamic:='true';
        S7_contact:='true'} :   BOOL := 0; // Linkable Input RESET
CSF  {S7_dynamic:='true'} :     BOOL := 0; // Control System Fault 1=External Error
FB_OPEN  {S7_qc:='true';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // Feedback: 1=OPEN
QC_FB_OPEN : BYTE := 16#80;                // Quality Code for FB_OPEN
FB_CLOSE     {S7_qc:='true';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;     // Feedback: 1=CLOSE
QC_FB_CLOSE : BYTE := 16#80;               // Quality Code for FB_CLOSE
QC_QCONTROL_I : BYTE := 16#80;             // Quality Code for Input QCONTROL
NO_FB_OP     {S7_visible:='false'} :    BOOL := 0; // 1=No Feedback OPEN present
NO_FB_CL     {S7_visible:='false'} :    BOOL := 0; // 1=No Feedback CLOSE present
```

```
MONITOR  {S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='Monitoring=Off';
         S7_string_1:='Monitoring=On'} :     BOOL := 1; // Select: 1=Monitoring ON,
                                                        // 0=Monitoring OFF
NOMON_OP     {S7_visible:='false'} :  BOOL := 0; // 1=No  Monitoring OPEN
NOMON_CL     {S7_visible:='false'} :  BOOL := 0; // 1=No  Monitoring CLOSE
OP_OP_EN     {S7_visible:='false'} :  BOOL := 1; // Enable: 1=Operator may input OPEN
CL_OP_EN     {S7_visible:='false'} :  BOOL := 1; // Enable: 1=Operator may input CLOSE
MANOP_EN     {S7_visible:='false'} :  BOOL := 1; // Enable: 1=Operator may input MANUAL
AUTOP_EN     {S7_visible:='false'} :  BOOL := 1; // Enable: 1=Operator may input AUTO
LIOP_SEL     {S7_contact:='true'}  :  BOOL := 0; // Select: 1=Linking, 0=Operator Active
AUT_L    {S7_dynamic:='true';
         S7_contact:='true'} :   BOOL := 0;  // Linkable Input for MANUAL/AUTO Mode
TIME_MON     {S7_link:='false';
         S7_edit:='para';
         S7_m_c:='true';
         S7_shortcut:='Mon. Time';
         S7_unit:='s'} :     REAL := 3;           // Monitoring Time [s]
SAMPLE_T     {S7_visible:='false';
         S7_sampletime:='true'} :    REAL := 1;   // Sample Time [s]
MSG_EVID     {S7_visible:='false';
         S7_link:='false';
         S7_param :='false';
         S7_server:='alarm_archiv';S7_a_type:='alarm_8p'} :  DWORD := 0; // Message ID
BA_EN    {S7_visible:='false';
         S7_m_c:='true'} :   BOOL := 0;          // Batch Enable
OCCUPIED     {S7_visible:='false';
         S7_m_c:='true'} :   BOOL := 0;          // Occupied by Batch
BA_ID    {S7_visible:='false';
         S7_m_c:='true'} :   DWORD := 0;         // Batch ID
BA_NA    {S7_visible:='false';
         S7_m_c:='true'} :   STRING[32] := '';   // Batch Name
STEP_NO  {S7_visible:='false';
         S7_m_c:='true'} :   DWORD := 0;         // Batch Step Number
RUNUPCYC     {S7_visible:='false';
         S7_link:='false'} :     INT := 3;       // Lag: Number of Run Up Cycles
USTATUS  {S7_visible:='false'} :    WORD := 0;  // User STATUS Bits
END_VAR

VAR_IN_OUT
RESET    {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='0';
         S7_string_1:='Error=Reset'} :   BOOL := 0; // Operator Input Error Reset
MAN_OC   {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='Valve=Close';
         S7_string_1:='Valve=Open'} :    BOOL := 0; // Operator Input: 1=OPEN, 0=CLOSE
AUT_ON_OP    {S7_visible:='false';
         S7_link:='false';
         S7_m_c:='true';
         S7_string_0:='Mode=Manual';
         S7_string_1:='Mode=Auto'} :     BOOL := 0; // Operator Input Mode 1=AUTO,
                                                      // 0= MANUAL
AUX_PR04     {S7_visible:='false'} :    ANY;       // Auxiliary Value 4
AUX_PR05     {S7_visible:='false'} :    ANY;       // Auxiliary Value 5
AUX_PR06     {S7_visible:='false'} :    ANY;       // Auxiliary Value 6
AUX_PR07     {S7_visible:='false'} :    ANY;       // Auxiliary Value 7
AUX_PR08     {S7_visible:='false'} :    ANY;       // Auxiliary Value 8
AUX_PR09     {S7_visible:='false'} :    ANY;       // Auxiliary Value 9
AUX_PR10     {S7_visible:='false'} :    ANY;       // Auxiliary Value 10
END_VAR

VAR_OUTPUT
QERR     {S7_visible:='false';
         S7_m_c:='true'} :   BOOL := 1;      // 1=Error
QCONTROL     {S7_qc:='true';
         S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0;      // Control Output: 0=Standstill
QC_QCONTROL : BYTE := 16#80;                 // Quality Code Output for QCONTROL
QMON_ERR     {S7_dynamic:='true';
         S7_m_c:='true'} :   BOOL := 0;      // 1=Monitoring Error
QGR_ERR  {S7_dynamic:='true';
         S7_contact:='true'} :   BOOL := 0;  // 1=Group Error
QMAN_AUT     {S7_dynamic:='true';
         S7_contact:='true';
```

```
        S7_m_c:='true'} :   BOOL := 0;      // 1=AUTO, 0=MANUAL Mode
QOP_ERR  {S7_visible:='false';
        S7_dynamic:='true'} :    BOOL := 0; // 1=Operator Error
QOP_OP   {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // Status: 1=Operator enabled for "OPEN"
QCL_OP   {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // Status: 1=Operator enabled for "CLOSE"
QAUTOP   {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // Status: 1=Operator enabled for "AUTO"
QMANOP   {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // Status: 1=Oper. ena. for "MANUAL" Mode
QOPENING     {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // 1=Valve is Opening
QOPENED  {S7_dynamic:='true';
        S7_contact:='true';
        S7_m_c:='true'} :   BOOL := 0;      // 1=Valve is OPEN
QCLOSING     {S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // 1=Valve is Closing
QCLOSED  {S7_dynamic:='true';
        S7_contact:='true';
        S7_m_c:='true'} :   BOOL := 0;      // 1=Valve is CLOSED
QMSG_ERR     {S7_visible:='false';
        S7_dynamic:='true'} :   BOOL := 0;  // 1=Message ERROR
QMSG_SUP     {S7_visible:='false';
        S7_dynamic:='true';
        S7_m_c:='true'} :   BOOL := 0;      // 1=Message Suppression Active
MSG_STAT     {S7_visible:='false';
        S7_dynamic:='true'} :   WORD := 0;  // Message: STATUS Output
MSG_ACK  {S7_visible:='false';
        S7_dynamic:='true'} :   WORD := 0;  // Message: ACK_STATE-output
VSTATUS  {S7_visible:='false';
        S7_m_c:='true'} :   DWORD := 0;     // Status-word
END_VAR

CONST
C_OOS := 0;                 // 1= Out of Service
C_V_LOCK := 0;              // 1=Lock to SAVE position
C_VL_OPEN := 0;             // 1=Lock to OPEN
C_VL_CLOSE := 0;            // 1=Lock to CLOSE
C_AUTO_OC := 0;             // AUTO Mode:1=Open, 0=Close
C_SS_POS := 0;              // Safe Position. 1=Open, 0=Close
C_START_SS := 1;            // 1=Start with Safe State Position and Manual Mode
C_FAULT_SS := 1;            // 1=In Case of Fault: Safe State Position
C_RESET := 0;               // Operator Input Error Reset
C_L_RESET := 0;             // Linkable Input RESET
C_CSF := 0;                 // Control System Fault 1=External Error
C_FB_OPEN := 0;             // Feedback: 1=OPEN
C_QC_FB_OPEN := 16#80;      // Quality Code for FB_OPEN
C_FB_CLOSE := 0;            // Feedback: 1=CLOSE
C_QC_FB_CLOSE := 16#80;     // Quality Code for FB_CLOSE
C_QC_QCONTROL_I := 16#80;   // Quality Code for Input QCONTROL
C_NO_FB_OP := 0;            // 1=No Feedback OPEN present
C_NO_FB_CL := 0;            // 1=No Feedback CLOSE present
C_MONITOR := 1;             // Select: 1=Monitoring ON, 0=Monitoring OFF
C_NOMON_OP := 0;            // 1=No  Monitoring OPEN
C_NOMON_CL := 0;            // 1=No  Monitoring CLOSE
C_MAN_OC := 0;              // Operator Input: 1=OPEN, 0=CLOSE
C_OP_OP_EN := 1;            // Enable 1=Operator may input OPEN
C_CL_OP_EN := 1;            // Enable: 1=Operator may input CLOSE
C_AUT_ON_OP := 0;           // Operator Input Mode 1=AUTO, 0= MANUAL
C_MANOP_EN := 1;            // Enable: 1=Operator may input MANUAL
C_AUTOP_EN := 1;            // Enable: 1=Operator may input AUTO
C_LIOP_SEL := 0;            // Select: 1=Linking, 0=Operator Active
C_AUT_L := 0;               // Linkable Input for  MANUAL/AUTO Mode
C_TIME_MON := 3;            // Monitoring Time [s]
C_SAMPLE_T := 1;            // Sample Time [s]
C_AUX_PR04 := 0;            // Auxiliary Value 4
C_AUX_PR05 := 0;            // Auxiliary Value 5
C_AUX_PR06 := 0;            // Auxiliary Value 6
C_AUX_PR07 := 0;            // Auxiliary Value 7
C_AUX_PR08 := 0;            // Auxiliary Value 8
C_AUX_PR09 := 0;            // Auxiliary Value 9
C_AUX_PR10 := 0;            // Auxiliary Value 10
C_MSG_EVID := 0;            // Message ID
C_BA_EN := 0;               // Batch Enable
```

```
C_OCCUPIED := 0;             // Occupied by Batch
C_BA_ID := 0;                // Batch ID
C_BA_NA := '';               // Batch Name
C_STEP_NO := 0;              // Batch Step Number
C_RUNUPCYC := 3;             // Lag: Number of Run Up Cycles
C_USTATUS := 0;              // User STATUS Bits
C_QERR := 1;                 // 1=Error
C_QCONTROL := 0;             // Control Output: 0=Standstill
C_QC_QCONTROL := 16#80;      // Quality Code Output for QCONTROL
C_QMON_ERR := 0;             // 1=Monitoring Error
C_QGR_ERR := 0;              // 1=Group Error
C_QMAN_AUT := 0;             // 1=AUTO, 0=MANUAL Mode
C_QOP_ERR := 0;              // 1=Operator Error
C_QOP_OP := 0;               // Status: 1=Operator enabled for "OPEN"
C_QCL_OP := 0;               // Status: 1=Operator enabled for "CLOSE"
C_QAUTOP := 0;               // Status: 1=Operator enabled for "AUTO"
C_QMANOP := 0;               // Status: 1=Oper. ena. for "MANUAL" Mode
C_QOPENING := 0;             // 1=Valve is Opening
C_QOPENED := 0;              // 1=Valve is OPEN
C_QCLOSING := 0;             // 1=Valve is Closing
C_QCLOSED := 0;              // 1=Valve is CLOSED
C_QMSG_ERR := 0;             // 1=Message ERROR
C_QMSG_SUP := 0;             // 1=Message Suppression Active
C_MSG_STAT := 0;             // Message: STATUS Output
C_MSG_ACK := 0;              // Message: ACK_STATE-output
C_VSTATUS := 0;              // Status-word
END_CONST

// Static variables
VAR
sbRESTART: BOOL := 1;          // First run(Default is1)
 sb_SIG_1: BOOL    := FALSE;   // ALARM_8P signal 1 flag
 sb_SIG_2: BOOL    := FALSE;   // ALARM_8P signal 2 flag

//------------------------- OP_D.2 -----------------------------
sbAUT_ON_OP:   BOOL := C_AUT_ON_OP;  // Flag of old operating value for(I0) (0=MAN)

//------------------------- OP_D.1 -----------------------------
sbMAN_OC:  BOOL := C_MAN_OC;         // Flag of old operating value for(I0) (0=CLOSED)
sbD1_Q_OC: BOOL := FALSE;            // Q Ausgang

//------------------------- OP_TRIG.1  ---------------------------
sbL_RESET: BOOL := C_L_RESET;        // Flag for historical signal L_RESET (LINK_I)

//------------------------- Watchdog-----------------------------
sbOC:      BOOL := FALSE;            // Old values direction
srAKTZEIT: REAL := 0;                // Current time of the on-delay
siRUNUPCNT:    INT := 0;             // Counter for RUNUPCYC editing

//------------------------- ALARM_8P.1 ---------------------------
ALARM_8P_1:    ALARM_8P;           // Multi-instanced ALARM_8P
dwDUMMY: DWORD := 0;               // Stand-by
siBA_ID:   DWORD := 0;             // Old value BA_ID
sbyBA_NA:  ARRAY[1..32] OF BYTE := 32(0);

VSTATUS_LOC : DWORD := 16#0; // Local variable, into which the VSTATUS output is copied.
STEP_NO_LOC : DWORD;         // Local variable, into which the STEP_NO input is copied.
BA_ID_LOC : DWORD;           // Local variable, into which the BA_ID input is copied.

// Variables for STATUSWORT "VSTATUS" and "USTATUS"
VSTATUS_STR AT VSTATUS_LOC :  STRUCT
                                USTATUS : WORD;
                                VSTATUS_LOW_BIT_8 : BOOL;
                                VSTATUS_LOW_BIT_9 : BOOL;
                                VSTATUS_LOW_BIT_10 : BOOL;
                                VSTATUS_LOW_BIT_11 : BOOL;
                                VSTATUS_LOW_BIT_12 : BOOL;
                                VSTATUS_LOW_BIT_13 : BOOL;
                                VSTATUS_LOW_BIT_14 : BOOL;
                                VSTATUS_LOW_BIT_15 : BOOL;
                                VSTATUS_LOW_BIT_0 : BOOL;
                                VSTATUS_LOW_BIT_1 : BOOL;
                                VSTATUS_LOW_BIT_2 : BOOL;
                                VSTATUS_LOW_BIT_3 : BOOL;
                                VSTATUS_LOW_BIT_4 : BOOL;
                                VSTATUS_LOW_BIT_5 : BOOL;
                                VSTATUS_LOW_BIT_6 : BOOL;
                                VSTATUS_LOW_BIT_7 : BOOL;
```

```
                                END_STRUCT;
END_VAR

// Temporary variables
VAR_TEMP
TOP_SI:     STRUCT
   EV_CLASS:    BYTE;
   EV_NUM:      BYTE;
   PRIORITY:    BYTE;
   NUM:         BYTE;
   TYP2_3:      BYTE;
   TYP1:        BYTE;
   ZI1:         WORD;
   ZI2_3:       DWORD;
   END_STRUCT;
START_UP_SI:    STRUCT
   EV_CLASS:    BYTE;
   EV_NUM:      BYTE;
   PRIORITY:    BYTE;
   NUM:         BYTE;
   TYP2_3:      BYTE;
   TYP1:        BYTE;
   ZI1:         WORD;
   ZI2_3:       DWORD;
   END_STRUCT;
ERR :       INT;        // Startup error
pbALARM:    BOOL;       // Call up ALARM_8P
pbM_SUP:    BOOL;       // Message suppression
//--------------------------- SEL_BOOL.1 -------------------------
pbSEL_BOOL_Q:       BOOL;   // SEL_BOOL 1 output
//--------------------------- OP_D.1 and 2 -----------------------
pbQOP_ERR:      BOOL;       // Temporary output value
//--------------------------- OP_TRIG.1   ------------------------
pbRESET:        BOOL;       // RESET (Ausgang Q)
//--------------------------- Watchdog ---------------------------
pbQMON_ERR:         BOOL;   // Watchdog error
pbVerfahren:        BOOL;   // Valve travel
//--------------------------- Feedback ---------------------------
pbFB_OPEN:      BOOL;
pbFB_CLOSE:     BOOL;
END_VAR
BEGIN
    // Write VSTATUS, resave STEP_NO as STEP_NO_LOC, resave BA_ID as BA_ID_LOC.
    // Supply of the VSTATUS output variable with the inputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_0 := OCCUPIED;
    VSTATUS_STR.VSTATUS_LOW_BIT_1 := BA_EN;
    // Supply of the VSTATUS output variable (HIGH BYTE) with the values
      // that come from the user via the input variable USTATUS (from external).
    VSTATUS_STR.USTATUS := USTATUS;
    //Resave STEP_NO as STEP_NO_LOC und resave BA_ID as BA_ID_LOC
    STEP_NO_LOC := STEP_NO; // Resave, due to output of STEP_NO_LOC in ALARM_8P
    BA_ID_LOC := BA_ID;     // Resave, due to output of BA_ID_LOC in ALARM_8P
    ERR := RD_SINFO (TOP_SI := TOP_SI, START_UP_SI := START_UP_SI); // Read out
                                                                    // start info.
    pbM_SUP := FALSE;
    IF TOP_SI.NUM = 100  OR sbRESTART THEN  // Startup or first run
        sbRESTART := FALSE;                 // No first run anymore
        QMON_ERR := C_QMON_ERR;             // No error
        pbALARM :=TRUE;                     // Initialization ALARM_8P
        IF START_SS  THEN                   // At Start SafeStatePosition?
            AUT_ON_OP := false;             // Manual mode
            sbAUT_ON_OP := false;           // Manual mode
            sbMAN_OC := SS_POS;             // Idle position
            MAN_OC := SS_POS;               // Write back idle position
            sbDI_Q_OC := SS_POS;            // Idle position
            srAKTZEIT := 0;                 // Initialize watchdogs time
            QMAN_AUT := C_QMAN_AUT;         // Manual
            QOPENING := C_QOPENING;         // No movement
            QCLOSING := C_QCLOSING;
            sbL_RESET := C_L_RESET;         // No RESET
            RESET := C_RESET;               // No RESET
            QCONTROL := C_QCONTROL;         // Idle position
            QC_QCONTROL := QC_QCONTROL_I;   // Describe quality output with values of
                                              // input.
            QMSG_ERR := C_QMSG_ERR;
            QMSG_SUP := C_QMSG_SUP;
            MSG_STAT := C_MSG_STAT;
            MSG_ACK  := C_MSG_ACK;
```

```
            QGR_ERR  := C_QGR_ERR;
            QCL_OP := C_QCL_OP;
            QOP_OP := C_QOP_OP;
            QOP_ERR := C_QOP_ERR;
            QMANOP := C_QMANOP;
            QAUTOP := C_QAUTOP;
        ELSE;
            IF (NOT AUT_ON_OP AND                   // Manual mode
                MONITOR)    THEN                    // Watchdog active
                IF NOT NO_FB_OP AND                 // Feedback OPEN available
                   FB_OPEN  THEN                    // Feedback available
                    MAN_OC := TRUE;                 // Track operator input
                    QCONTROL := (MAN_OC XOR SS_POS);
                    QC_QCONTROL := QC_QCONTROL_I;  // Describe quality output with
                                                      // values of input.
                    QOPENED := TRUE;
                    QOPENING := FALSE;
                    QCLOSED := FALSE;
                    QCLOSING := FALSE;
                END_IF;
                IF NOT NO_FB_CL AND                 // Feedback CLOSE available
                   FB_CLOSE    THEN                 // Feedback available
                    MAN_OC := FALSE;                // Track operator input
                    QCONTROL := (MAN_OC XOR SS_POS);
                    QC_QCONTROL := QC_QCONTROL_I;  // Describe quality output with
                                                      // values of input.
                    QOPENED := FALSE;
                    QOPENING := FALSE;
                    QCLOSED := TRUE;
                    QCLOSING := FALSE;
                END_IF;
            END_IF;
            QCL_OP := CL_OP_EN;
            QOP_OP := OP_OP_EN;
        END_IF;
    END_IF;
    IF TOP_SI.NUM = 100 THEN     // Initialization at startup
        siRUNUPCNT := RUNUPCYC;  // Save the value of the RUNUPCYC input
    ELSE;
        // VALVE algorithm
        pbQOP_ERR := FALSE;      // Optimistic initializations
        pbQMON_ERR := FALSE;
        pbVerfahren := FALSE;    // No OPEN/CLOSE operation
        // OP_D.2 (optimized)                          - 1 -
        IF LIOP_SEL THEN
            // Interconnection
            IF AUT_ON_OP = sbAUT_ON_OP THEN
                ;// No operation occurred
            ELSE; // Prohibited operator
                pbQOP_ERR := TRUE;
            END_IF;
            AUT_ON_OP := AUT_L;       // Write AUT_L back to operator input
                                         // (BTRACK is always1).
            QMAN_AUT := AUT_L;        // Interconnection
            QMANOP := FALSE;          // No operation allowed
            QAUTOP := FALSE;
        ELSE;        // Bedienung
            IF (sbAUT_ON_OP <> AUT_ON_OP) AND
               NOT ((AUT_ON_OP AND AUTOP_EN) OR (NOT(AUT_ON_OP) AND MANOP_EN)) THEN
                pbQOP_ERR := TRUE;          // Prohibited operator
                AUT_ON_OP := sbAUT_ON_OP;   // Write back old operating values
            ELSE;                           // No or allowed operation
                QMAN_AUT := AUT_ON_OP;
            END_IF;
            QMANOP := MANOP_EN;         // Copy the inputs to the outputs
            QAUTOP := AUTOP_EN;
        END_IF;
        sbAUT_ON_OP := AUT_ON_OP;       //Note old operating values
        //OP_TRIG.1 (optimized)                       - 2 -
        IF ((L_RESET AND (NOT sbL_RESET)) OR RESET) THEN   // Rising edge was for
                                                              // L_RESET or RESET.
            pbRESET := TRUE;            // RESET
        ELSE;
            pbRESET := FALSE;           // No RESET
        END_IF;
        sbL_RESET := L_RESET;           // Save previous values
        RESET := FALSE;                 // Reset operator-controllable input
        // Watchdog RESET Logic                        - 3 -
```

```
IF QMON_ERR  THEN                // QMON_ERR Flip-Flop
    IF NOT pbRESET THEN
        pbQMON_ERR := TRUE;
    ELSE;
        // pbQMON_ERR bleibt 0
        // pbRESET bleibt wirksam
        IF TIME_MON < SAMPLE_T  THEN // Reposition watchdog time
            srAKTZEIT := SAMPLE_T;
        ELSE;
            srAKTZEIT := TIME_MON;
        END_IF;
    END_IF;
ELSE;
    // RESET without error not effective
    pbRESET := FALSE;
END_IF;
IF V_LOCK OR VL_OPEN OR VL_CLOSE OR (pbQMON_ERR AND FAULT_SS)// SEL_BOOL.1 - 4 -
THEN
    pbSEL_BOOL_Q := SS_POS;          // Idle position
    IF (NOT V_LOCK) THEN
        IF VL_CLOSE THEN
            pbSEL_BOOL_Q := 0;       // Close valve
        ELSE;
            IF VL_OPEN THEN
                pbSEL_BOOL_Q := 1;  // Open valve
            END_IF;
        END_IF;
    END_IF;
ELSE;
    pbSEL_BOOL_Q := AUTO_OC;         // Automatic value
END_IF;
// OP_D.1 (optimized and expanded with LINK_ON calculation)           - 5 -
IF QMAN_AUT OR V_LOCK OR VL_OPEN OR VL_CLOSE OR (pbQMON_ERR AND FAULT_SS)
                                     // LINK_ON
THEN // Interconnection
    IF MAN_OC = sbMAN_OC THEN
        ; // Keine  Bedienung erfolgt
    ELSE;
        pbQOP_ERR := TRUE;             // Prohibited operator
    END_IF;
    IF sbD1_Q_OC <> pbSEL_BOOL_Q  THEN // Valve travel?
        pbVerfahren := TRUE;
        sbD1_Q_OC := pbSEL_BOOL_Q;     // LINK_I
    END_IF;
    MAN_OC := sbD1_Q_OC;    // Write output back to operator input
                              // (BTRACK is always1).
ELSE;   // Bedienung
    IF (sbMAN_OC <> MAN_OC) AND
       NOT ((MAN_OC AND OP_OP_EN) OR (NOT(MAN_OC) AND CL_OP_EN)) THEN
        pbQOP_ERR := TRUE;             // Prohibited operator
        MAN_OC := sbMAN_OC;
        // Write back old operating values
    ELSE;
        IF sbD1_Q_OC <> MAN_OC  THEN   // Valve travel?
            // Allowed operator
            pbVerfahren := TRUE;
            sbD1_Q_OC := MAN_OC;
        END_IF;
    END_IF;
END_IF;
sbMAN_OC := MAN_OC;    //Note old operating values
IF pbVerfahren         // Valve is traveled -> Reposition watchdog time
THEN
    IF TIME_MON < SAMPLE_T THEN
        srAKTZEIT := SAMPLE_T;
    ELSE;
        srAKTZEIT := TIME_MON;
    END_IF;
END_IF;
```

```
// Watchdog and position display                          - 6 -
IF NOT MONITOR  THEN                  // No feedback watchdogs
    IF (srAKTZEIT >= SAMPLE_T) THEN         // Timer is running
        srAKTZEIT := srAKTZEIT - SAMPLE_T; // Bring down time
        pbFB_OPEN := FALSE;
        pbFB_CLOSE := FALSE;
    ELSE;
        pbFB_OPEN := sbD1_Q_OC;
        pbFB_CLOSE := NOT sbD1_Q_OC;
    END_IF;
ELSE;
    IF NO_FB_OP  THEN   // No limit switch OPEN
        pbFB_OPEN := sbD1_Q_OC;
    ELSE;
        pbFB_OPEN := FB_OPEN;
    END_IF;
    IF NO_FB_CL  THEN   // No limit switch CLOSE
        pbFB_CLOSE := NOT sbD1_Q_OC;
    ELSE;
        pbFB_CLOSE := FB_CLOSE;
    END_IF;
END_IF;
IF MONITOR AND (((sbD1_Q_OC  AND NOT pbFB_OPEN) AND NOT NOMON_OP) OR
   ((NOT sbD1_Q_OC AND NOT pbFB_CLOSE) AND NOT NOMON_CL) OR
   (pbFB_OPEN AND pbFB_CLOSE) AND NOT (NOMON_OP AND NOMON_CL))THEN
    IF (srAKTZEIT < SAMPLE_T) THEN
        pbQMON_ERR := TRUE;
    END_IF;
    IF (srAKTZEIT >= SAMPLE_T) THEN         // Timer is running
        srAKTZEIT := srAKTZEIT - SAMPLE_T; // Bring down time
    END_IF;
ELSE;
    IF (sbD1_Q_OC  AND pbFB_OPEN) OR
       (NOT sbD1_Q_OC AND pbFB_CLOSE) THEN
        srAKTZEIT := 0;    // End position reached-> Reset watchdog time
    END_IF;
END_IF;
QMON_ERR := pbQMON_ERR;
IF QMAN_AUT OR V_LOCK OR VL_OPEN OR VL_CLOSE OR (QMON_ERR AND FAULT_SS) THEN
    // No OPEN/CLOSE operation allowed
    QOP_OP := FALSE;
    QCL_OP := FALSE;
ELSE;
    // Copy the inputs to the outputs
    QOP_OP := OP_OP_EN;
    QCL_OP := CL_OP_EN;
END_IF;
// Position displays
QOPENED  := sbD1_Q_OC AND pbFB_OPEN AND NOT pbFB_CLOSE;
QCLOSED  := NOT sbD1_Q_OC  AND pbFB_CLOSE AND NOT pbFB_OPEN;
IF QOPENED THEN
    QOPENING := FALSE;
ELSE;
    QOPENING := sbD1_Q_OC AND NOT QMON_ERR;
END_IF;
IF QCLOSED THEN
    QCLOSING := FALSE;
ELSE;
    QCLOSING := NOT sbD1_Q_OC  AND NOT QMON_ERR;
END_IF;
QOP_ERR  := pbQOP_ERR;              // Replace outputs
QGR_ERR  := QMON_ERR OR CSF;        // Calculate group monitor error
IF FAULT_SS AND (NOT(V_LOCK OR VL_OPEN OR VL_CLOSE))
THEN // Safe State Position at error?
    QCONTROL := (sbD1_Q_OC XOR SS_POS) AND NOT QMON_ERR;
    QC_QCONTROL := QC_QCONTROL_I;   // Describe quality output with values of
                                      // input
ELSE;
    QCONTROL := (sbD1_Q_OC XOR SS_POS);
    QC_QCONTROL := QC_QCONTROL_I;   // Describe quality output with values of
                                      // input
END_IF;
IF siRUNUPCNT = 0  THEN             // Initialize alarms
    // pbALARM := (sb_SIG_1 <> QMON_ERR) OR (sb_SIG_2 <>  CSF);
    pbALARM :=TRUE;                 // Initialization ALARM_8P
ELSE;
```

```
            pbALARM :=FALSE;                // Initialization no ALARM
            siRUNUPCNT := siRUNUPCNT - 1;
            pbM_SUP := TRUE;
        END_IF;
    END_IF;
    IF pbALARM THEN    // Call up ALARM_8P
        IF siBA_ID <> BA_ID_LOC THEN
            // STRING variables may not be interconnected to ALARM8_P as auxiliary
              // process values, therefore transferred in ARRAY OF BYTE.
            FOR ERR := 1 TO 32
             DO
                sbyBA_NA[ERR] := 0;    // Delete array as default
            END_FOR;
            ERR := BLKMOV (SRCBLK:= BA_NA,DSTBLK:=sbyBA_NA);
            siBA_ID := BA_ID_LOC;      // Save modified BA_ID
        END_IF;

        // Call up ALARM_8P with lock logic (MSG_LOCK)
        ALARM_8P_1(    EN_R := TRUE,   // Update the ACKL_STATE output
                       ID := 16#EEEE,  // PMC communication channel
           EV_ID:= MSG_EVID,
           SIG_1:= QMON_ERR,
           SIG_2:= CSF,
           SIG_3:= 0,
           SIG_4:= 0,
           SIG_5:= 0,
           SIG_6:= 0,
           SIG_7:= 0,
           SIG_8:= 0,
           SD_1 := sbyBA_NA,
           SD_2 := STEP_NO_LOC,
           SD_3 := BA_ID_LOC,
           SD_4 := AUX_PR04,
           SD_5 := AUX_PR05,
           SD_6 := AUX_PR06,
           SD_7 := AUX_PR07,
           SD_8 := AUX_PR08,
           SD_9 := AUX_PR09,
           SD_10 := AUX_PR10);
        QMSG_ERR := ALARM_8P_1.ERROR;
        MSG_STAT := ALARM_8P_1.STATUS;
        MSG_ACK := ALARM_8P_1.ACK_STATE;
        sb_SIG_1:= QMON_ERR;     // Note historical signals
        sb_SIG_2:= CSF;
    END_IF;
    IF (MSG_STAT = 21) THEN      // Block locked
        pbM_SUP := TRUE;
    END_IF;
    QMSG_SUP := pbM_SUP;
    QERR := NOT OK;              // Note negated OK-Flag result in the block

    //Power supply of the VSTATUS output variable with the outputs.
    VSTATUS_STR.VSTATUS_LOW_BIT_3 := QMAN_AUT;
    VSTATUS_STR.VSTATUS_LOW_BIT_7 := QMON_ERR;
    VSTATUS_STR.VSTATUS_LOW_BIT_8 := V_LOCK;
    VSTATUS_STR.VSTATUS_LOW_BIT_9 := QOPENED;
    VSTATUS_STR.VSTATUS_LOW_BIT_10 := QCLOSED;
    VSTATUS_STR.VSTATUS_LOW_BIT_11 := QOPENING;
    VSTATUS_STR.VSTATUS_LOW_BIT_12 := QCLOSING;
    VSTATUS_STR.VSTATUS_LOW_BIT_14 := QMSG_SUP;
    VSTATUS_STR.VSTATUS_LOW_BIT_15 := OOS;
    VSTATUS := VSTATUS_LOC;
END_FUNCTION_BLOCK
```

Glossary

A

Asynchronous OBs

Asynchronous OBs (organization blocks) are called by the operating system of the CPU when asynchronous events occur (for example errors).

AUTHOR

➔ *Block attribute*:

When a block library is used, this contains the library name. It is also used to identify the help file belonging to the library.

B

BATCH *flexible*

A program package for creating complex recipe controls for the entire range from small to large-scale applications.

Block attribute

Using the block attributes (➔ *FUNCTION_BLOCK*, ➔ *TITLE*, ➔ *List of system attributes*, ➔ *AUTHOR*, ➔ *NAME*, ➔ *VERSION*, ➔ *FAMILY*, ➔ *KNOW_HOW_PROTECT*) of the block, you can influence the object properties of your block.

Block header

1. Section of the ➔ *PLC block* with administrative information (➔ *block attributes*).
2. The upper part of the block in the graphic representation in CFC containing the name (block type, block name), comment and the task assignment (run-time property).

Block icon

Symbolic display of the most important information of a ➔ *PLC block*. The corresponding ➔ *faceplate* can be opened with the block icon.

C

CFC

Continuous Function Chart

1. This is a function chart on which blocks can be placed, interconnected, and assigned parameters. A CFC chart consists of 1 to 26 chart partitions each with 6 sheets.
2. Editor for plant-oriented graphic configuration of automation tasks. Using the CFC editor, a complete software structure (CFC chart) can be created from ready-made blocks.

CNT file

Optional part of an online help system. The CNT file contains the contents of the online help system.

Code section

Part of a block containing the algorithm of the block.

D

Declaration section

Part of a block defining the interface of the block and the data it uses internally.

F

Faceplate

Graphic representation of all the elements of a PLC block intended for operator control and monitoring on an OS.

FAMILY

➔ *Block attribute*:

When a block library is used, this contains a common name for a subset of the blocks. FAMILY and ➔ NAME form part of the key for locating the help text of a block in the online help system.

Function (FC)

Specified in IEC 1131–3 as a software unit that provides a single result when it executes (can also be a structured data type) and does not have memory for storing data. The major difference between an FC and an FB is the absence of data storage.

FUNCTION_BLOCK

➔ *Block attribute*:

Contains the symbolic name of the block. This is used to display the name of the block in the SIMATIC Manager and in the CFC chart.

Function block (FB)

According to IEC 1131-3, a function block is a logic block with static data. Using FBs, parameters can be passing in the user program. This makes function blocks suitable for programming frequently repeated complex functions, such as controllers, mode selection. Since the FB has a memory (instance data block), its parameters, for example outputs, can be accessed at any point in the user program.

G

Global script

Within ➔ *WinCC*, global script is the generic term for C functions created by the user that can be used throughout a project and in multiple projects.

Graphics Designer

Graphic editor in ➔ *WinCC* for creating faceplates.

H

HID

Higher-level identifier: This is made up of the name of the plant hierarchy folders that are selected to form part of the name and of the CFC chart and the block in the CFC chart.

I

Initial start

The first time that a block is executed following its instantiation. Following this first execution, the parameters and modes of the block are in a defined state.

K

KNOW_HOW_PROTECT

➔ *Block attribute*:

When this attribute is set, it protects the algorithm of the block from being viewed or modified unless the source file is in the same program.

L

Library

Software package with ➔ *PLC blocks* and/or ➔ *faceplates* grouped according to common features.

M

Message list

From within the run-time system of ➔ *WinCC*, it is possible to display and edit lists of messages. The messages displayed in the lists relate only the currently active project.

Monitoring

Part of the functions of an OS allowing visualization of the process parameters and statuses in various forms (numeric, graphic).

Multiple instance block

A block made up of several blocks. Its instance (data storage) contains the instances (data storage) of the FBs that are called in it.

N

NAME

➔ *Block attribute*:

Contains the symbolic name of the block; identical to ➔ FUNCTION_BLOCK. NAME and ➔ FAMILY form part of the key for locating the help text of a block in the online help system.

O

OK flag

The OK flag is a system-internal variable. If an error occurs during execution of an option, for example overflow with arithmetic operations, the OK flag is changed by the system and passed to the ENO block output.

Operator authorization

The right assigned to the operator to modify the ➔ *PLC block parameter.*

Operator control

Procedure in which the operator changes the value or status of a block. Generally, such operations involve input at the operator station which is then checked and passed to the block on the PLC. Here, the instruction is checked again before is assigned to the block since it is possible that process conditions have changed since the instruction was sent from the OS and received at the PLC.

P

PLC block

An object belonging to a library or a block structure that contains part of the S7 user program. The blocks that are executed in the CPU of a PLC are known as PLC blocks.

Prototype picture

Prototype pictures are used by ➔ *WinCC* to reuse picture components that have already been configured. The prototype picture technique makes use of template pictures that can be included more than once in one or more parent pictures. A prototype is simply a template that only "comes to life" in a real object. An object based on a template is created by instantiation. Several instances (on other words real objects) can be created from a template.

R

Registration file

And ASCII file (.reg) containing all the information such as path, MAP IDs for online help systems, for example to enter a block in the WINDOWS NT registry. Using this registration file, the online help for a selected block can be started in the required language in CFC or in the SIMATIC Manager.

S

SCL

Higher-level programming language for formulating solutions to technological tasks in SIMATIC S7 (similar to PASCAL) complying with the ST (structured text) language specified in IEC 1131–3.

Split Screen Wizard

Component of ➔ *WinCC*: This initializes the monitor and display settings on the OS.

Standard view

➔ *View* of a ➔ *faceplate* in which the most important values of the corresponding ➔ *PLC block* are visualized.

Start info

The start information is part of an organization block (OB). It informs the S7 user in detail of the event that triggered the OB call.

STL

Statement List: Statement List is a textual programming language complying with IEC 1131-3 and resembling machine code.

System attributes for blocks

Special attributes that prepare the ➔ *PLC block* for the connection to the OS or influence the installation of a block in a CFC chart.

System attributes for parameters

Special attributes that influence the display of the parameter in the ➔ *faceplate* or its handling in the CFC chart.

T

TITLE

➔ *Block attribute*:

This information is not evaluated in PCS 7, however, it is displayed in the SIMATIC Manager in the object properties of the block in the comment field.

Trend view

➔ *View* of a ➔ *faceplate* in which the most important values of the corresponding ➔ *PLC block* are visualized over time.

U

UDT

➔ *User-defined data types*

User-defined data types (UDT)

User-defined data types are special data structures that can be used in the entire CPU program after they have been defined. They cam be used like elementary or complex data types in the variable declaration of logic blocks (FCs, FBs, OBs) or as templates for creating data blocks with the same data structure.

V

VERSION

➔ *Block attribute*:

Contains the version number of the block

View

View of a block in which certain values of a PLC block are displayed (for example, trend view, alarm view, standard view etc.).

W

WinCC

Windows Control Center: A software package for plant-oriented graphic development of ➔ *faceplates* and for operator control and monitoring of the PLC.

Index

I

L

M

N

O

P

R

S

T

V

W