SIEMENS

SIMATIC

Programming Instructions, Creating Blocks for PCS 7

Manual

Edition 01/2001
A5E00092264-01

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 Automatisierungs- und Antriebstechnik
Geschaeftsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D- 90327 Nuernberg

©Siemens AG 2001
Technical data subject to change.

Excellence in Automation & Drives: Siemens

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

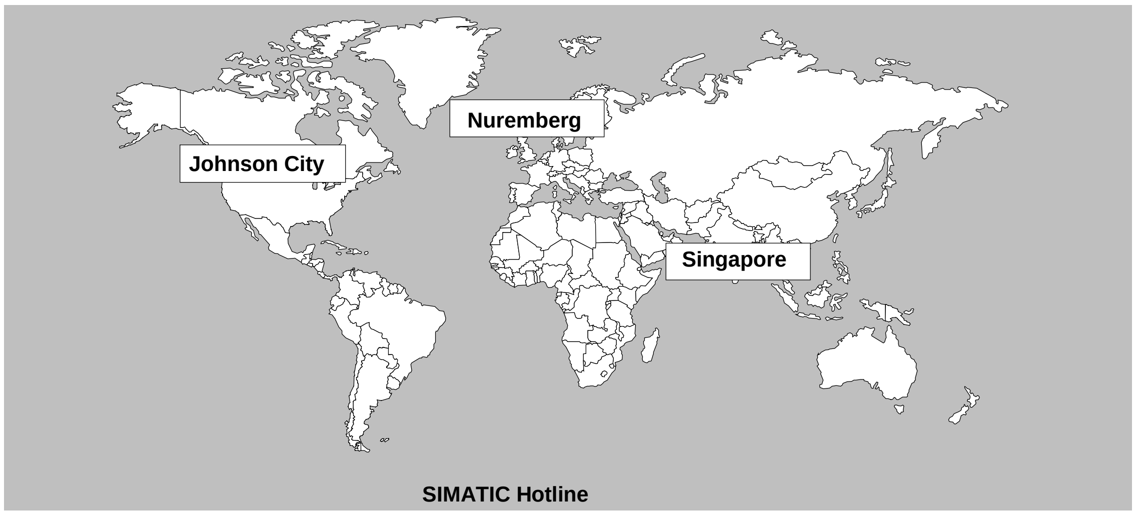
These programming instructions provide you with an overview of the individual components of a PCS 7-compliant block. The order in which they are introduced is the same order you would follow to develop function blocks and faceplates.

- 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 1.10 contains a printout of the entire sample PLC block.

Customer Support, Technical Support

Open round the clock, worldwide:



Worldwide (Nuremberg) **Technical Support**	**Worldwide** (Nuremberg) **Technical Support**	
(FreeContact) Local time: Mo.-Fr. 7:00 to 17:00 Phone: +49 (180) 5050 222 Fax: +49 (180) 5050 223 E-mail: techsupport@ ad.siemens.de GMT: +1:00	(charged, only with SIMATIC Card) Local time: Mo.-Fr. 0:00 to 24:00 Phone: +49 (911) 895-7777 Fax: +49 (911) 895-7001 GMT: +01:00	
Europe / Africa (Nuremberg) **Authorization**	**America** (Johnson City) **Technical Support and Authorization**	**Asia / Australia** (Singapore) **Technical Support and Authorization**
Local time: Mo.-Fr. 7:00 to 17:00 Phone: +49 (911) 895-7200 Fax: +49 (911) 895-7201 E-mail: authorization@ nbgm.siemens.de GMT: +1:00	Local time: Mo.-Fr. 8:00 to 19:00 Phone: +1 423 461-2522 Fax: +1 423 461-2289 E-mail: simatic.hotline@ sea.siemens.com GMT: -5:00	Local time: Mo.-Fr. 8:30 to 17:30 Phone: +65 740-7000 Fax: +65 740-7001 E-mail: simatic.hotline@ sae.siemens.com.sg GMT: +8:00
English and German are spoken on all hotlines. On the authorization hotline, French, Italian and Spanish are also spoken.		

SIMATIC Customer Support Online Services

The SIMATIC Customer Support team provides you with comprehensive additional information on SIMATIC products in its online services:

- You can obtain general current information:
  - on the **Internet** at http://www.ad.siemens.de/simatic
- Current Product Information leaflets and downloads which you may find useful for your product are available:
  - on the **Internet** at http://www.ad.siemens.de/simatic-cs
  - Via the **Bulletin Board System** (BBS) in Nuremberg (*SIMATIC Customer Support Mailbox)* number +49 (911) 895-7100.

  To access the mailbox, use a modem with V.34 (28.8 kilobauds) capability whose parameters you should set as follows: 8, N, 1, ANSI, or dial in using ISDN (x.75, 64 kbit/s).
- You can find your local contact for Automation & Drives in our contacts database:
  - On the **Internet** at http://www3.ad.siemens.de/partner/search.asp

Contents

1 Creating PLC Blocks

1.1 Requirements and Previous Experience

The blocks described here are intended for use with PCS 7 V5.1 and higher on an S7-4xx CPU. To create the blocks, you require the following software packages:

- STEP 7 standard, V5.1 or higher
- SCL Compiler, V5.1 or higher
- CFC, V5.1 or higher

PLC 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_CONTA.SCL**" (German) and "**S7_CONTB.SCL**" (English) with PCS7TOOLS and installed in ...\STEP7\Examples\ZDt25_01 or . ...\ZEn25_01 respectively.

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

- 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.

Before you compile, make sure that the "OP_A_LIM" block (FB46) 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\Technological Blocks" library.

- 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 **FB501** is now located in the block folder of your project.

1.2 Structure of a PLC Block

A PLC 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 (FB501). The individual elements of the block are explained in greater detail in the sections shown in brackets.

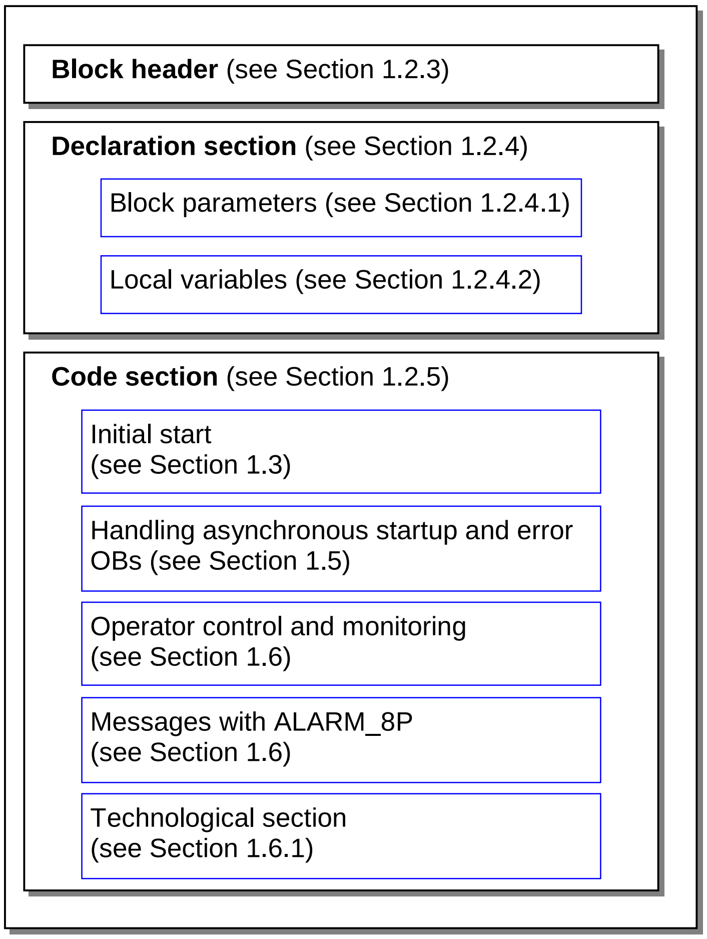

Block header (see Section 1.2.3)
Declaration section (see Section 1.2.4)
Block parameters (see Section 1.2.4.1)
Local variables (see Section 1.2.4.2)
Code section (see Section 1.2.5)
Initial start (see Section 1.3)
Handling asynchronous startup and error OBs (see Section 1.5)
Operator control and monitoring (see Section 1.6)
Messages with ALARM_8P (see Section 1.6)
Technological section (see Section 1.6.1)

Figure 1-1: Structure of the "CONTROL" (FB501) 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 PLC 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 PLC. 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 PLC block, the interface to the user (parameter names, comments etc.) should be in English. You can develop the blocks themselves in any language. The description and the samples relate to "English".

Selecting the Language

If you want to put your blocks together in a library (see also Chapter 4), the language must also be set to English so that the folders in your library have PCS 7-compliant names (**Sources**, **Symbols** 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 Symbol Table

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

To make this entry, open the symbol table in the S7 program by double-clicking "Symbols". Enter the symbolic name (here: "CONTROL") in the "Symbol" column and assign an FB number to it in the "Address" column (here: FB 501).
See also Section 1.10, Naming Conventions and Numeric Range.

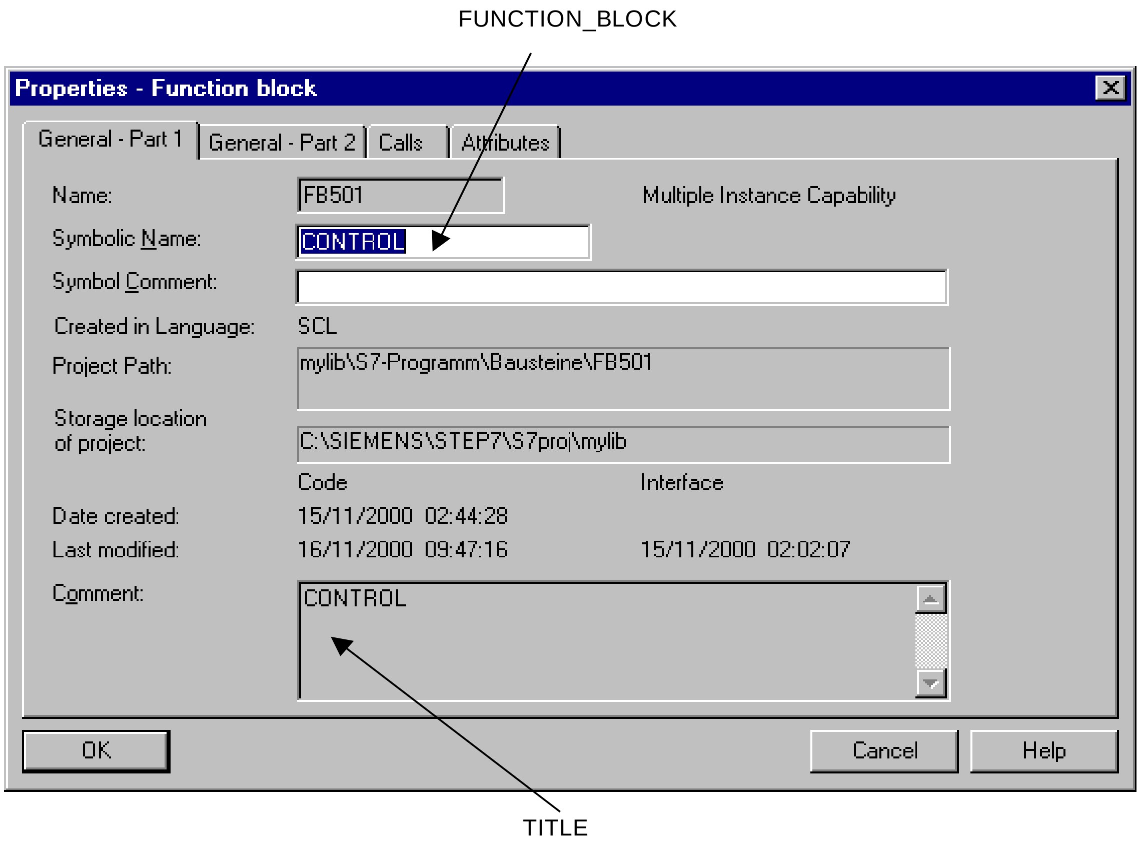
1.2.3 Block Header

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
  PLC block.

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

   If the manipulated variable 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 manipulated variable 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 BATCH flexible 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:                              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:     '0.01'
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.



Name: FB501

Multiple Instance Capability

Symbolic Name: CONTROL

Symbol Comment:

Created in Language: SCL

Project Path: mylib\S7-Programm\Bausteine\FB501

Storage location of project: C:\SIEMENS\STEP7\S7proj\mylib

	Code	Interface
Date created:	15/11/2000 02:44:28	
Last modified:	16/11/2000 09:47:16	15/11/2000 02:02:07

Comment: CONTROL

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

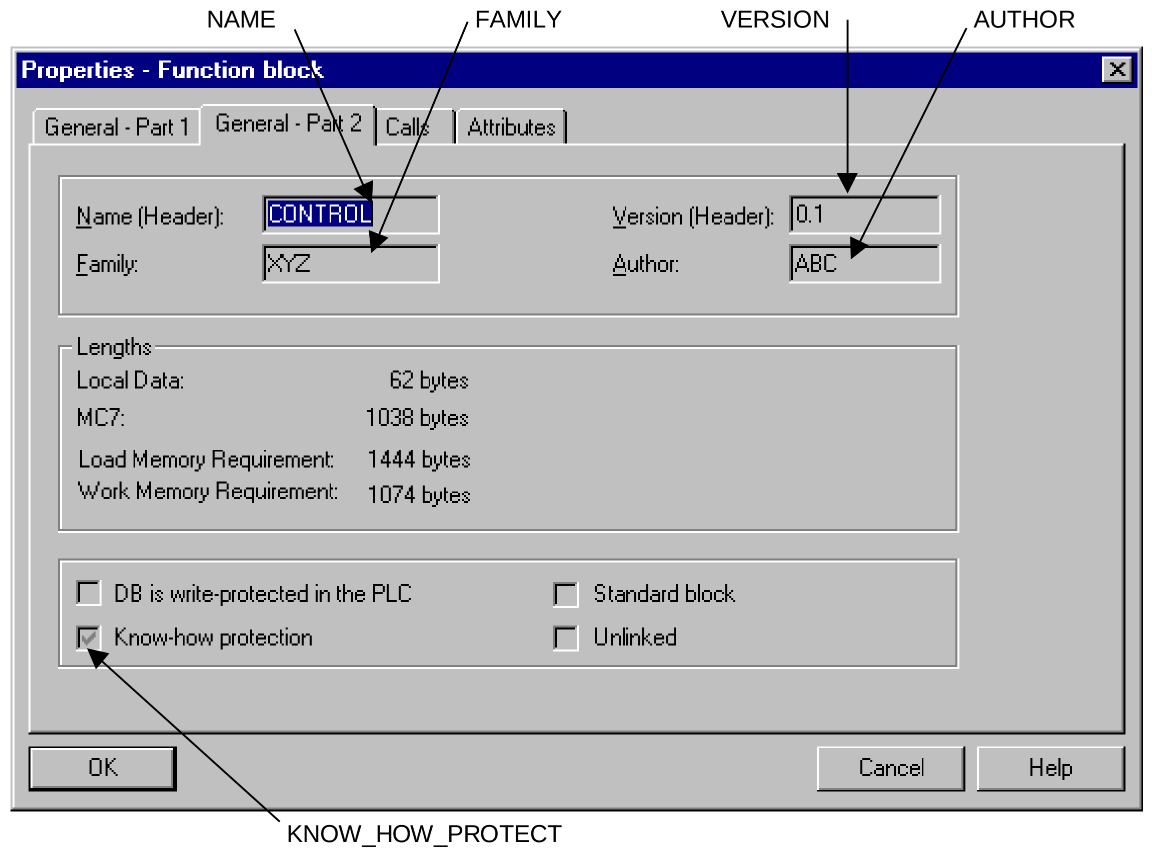
- **FUNCTION_BLOCK**

  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 symbol 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 the same name as for FUNCTION_BLOCK.



Name (Header): CONTROL
Family: XYZ
Version (Header): 0.1
Author: ABC

Lengths

Local Data:	62 bytes
MC7:	1038 bytes
Load Memory Requirement:	1444 bytes
Work Memory Requirement:	1074 bytes

☐ DB is write-protected in the PLC
☑ Know-how protection
☐ Standard block
☐ Unlinked

Figure 1-3: Object Properties of the Block (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 key 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 put your blocks together in a separate library and want to arrange the blocks in various groups within the library, 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 key for locating the help text of the block in the help file.

- **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 put your blocks together to form a 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**

  Using this attribute, you can protect the algorithm and the attributes of the block from being viewed and modified. 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.

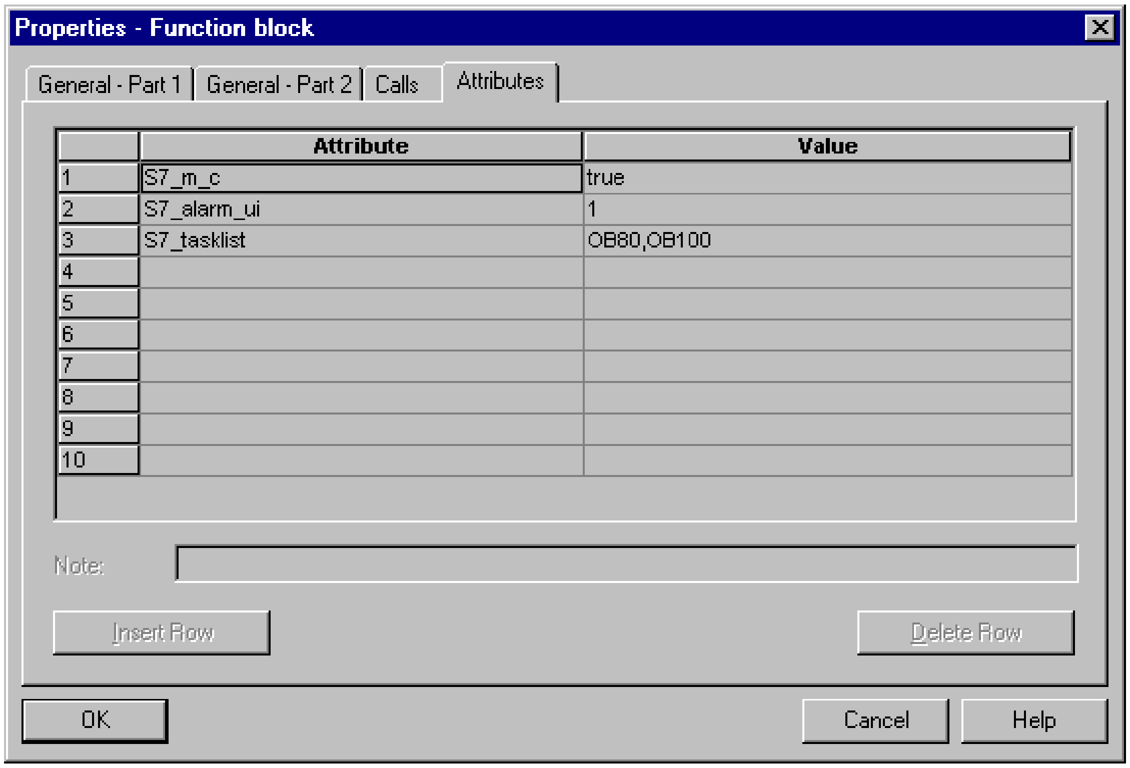
Table 1-1: System Attributes for PCS 7-Compliant Blocks

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.	false

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).



	Attribute	Value
1	S7_m_c	true
2	S7_alarm_ui	1
3	S7_tasklist	OB80,OB100
4		
5		
6		
7		
8		
9		
10		

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 or
  - You want to be able to influence the display of a faceplate on the OS using parameters (for example limits for the displayed ranges) or
  - You want to control parameters for test purposes when working in CFC or
  - You want to set parameters to generate messages (message event ID of the ALARM_8P block)

- **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 or
  - You want to monitor parameters at the OS or
  - 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 (PLC) 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 respon se	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_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

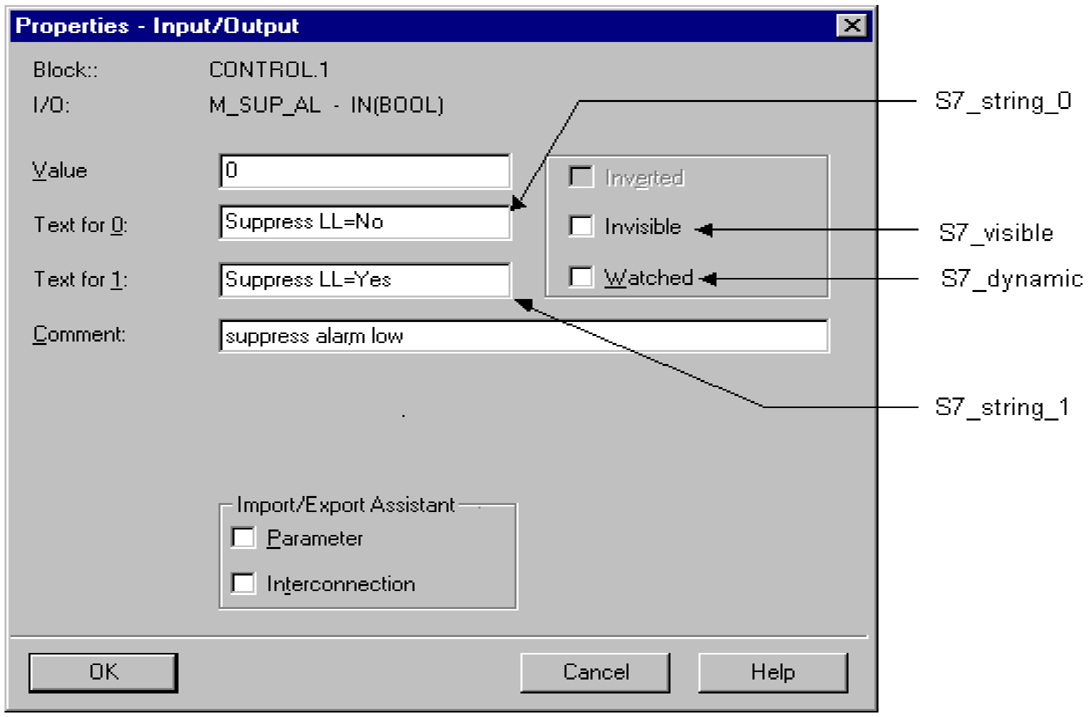
When using the system attributes "S7_string_0" and "S7_string_1", remember the following points:

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 PLC. 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".

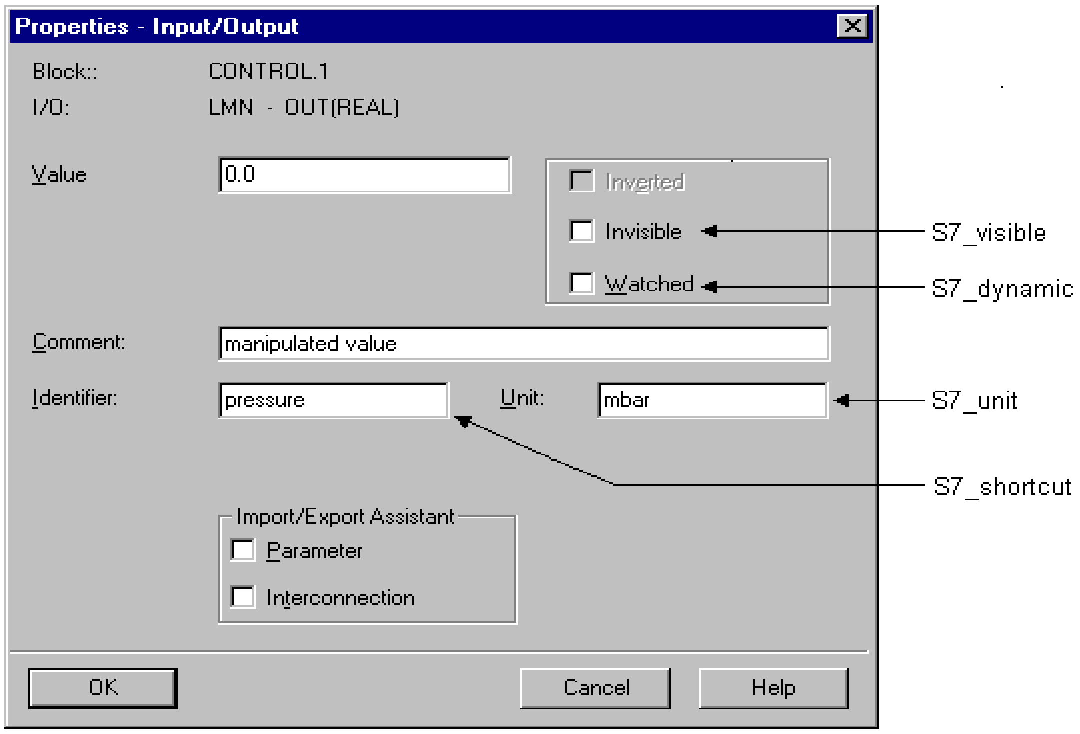
In CFC, the system attributes are displayed in the object properties of the I/O and can also be modified there. The following screenshots show the object properties of parameters of the data type BOOL and parameters that are not of the data type BOOL (Figure 1-6). The screenshots include arrows indicating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


Properties - Input/Output

Block:: CONTROL.1
I/O: M_SUP_AL - IN(BOOL)

Value: 0
Text for 0: Suppress LL=No
Text for 1: Suppress LL=Yes
Comment: suppress alarm low

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


Properties - Input/Output

Block:: CONTROL.1
I/O: LMN - OUT(REAL)

Value: 0.0
Comment: manipulated value
Identifier: pressure
Unit: mbar

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)

  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)

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

  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 PLC)
      }           :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 PLC)
      }
                  :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

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 BATCH flexible
  STEP_NO {S7_visible := 'false';
      S7_m_c   := 'true'}  :WORD;     // 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[16] := '';  // 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
 PV_IN {S7_dynamic:='true';
    S7_m_c:='true';
    S7_unit:='%'} :   REAL := 0;     // Process value (to AUX_PR04 of message)

 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 make parameters of the called block visible to the outside world and interconnectable, you must copy the parameters 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 SFC6 (RD_SINFO) or SFB0 (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 PLC 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 OB80 (time error).

Table 1-4: Assignment of the Elements of the TOP_SI Startup Information to the Temporary Variables of OB80

TOP_SI / STARTUP_SI		**OB80**	
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 (OB101).

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 Technological Blocks" 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 PLC 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 PLC block and evaluated later by the PLC 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 PLC and on the OS.

For binary operator input, you can use the OP_D (FB48), OP_D3 (FB49) and OP_TRIG (FB50) blocks of the PCS 7 "Technological Blocks" library (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
    SP_OP := OP_A_LIM_1.U;     // Write back setpoint

    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 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

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. This also applies to messages. To call the PCS 7 message dialog, 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 PLCs and operator stations. The numbers for individual messages required by WinCC are derived from this message number during data transfer.

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:          '0.01'
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. If no further action is taken, these messages are handled as process control messages 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)

 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:= 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
      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 := PV_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 PLC 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';
     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)
...
  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 (LMN > H_ALM) THEN   // If the manipulated variable violates the high alarm limit
     QH_ALM  := 1;     // set error output
     QL_ALM  := 0;     // reset error output

   ELSIF (LMN < L_ALM) THEN  // If the manipulated variable 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:= 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 FB501 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

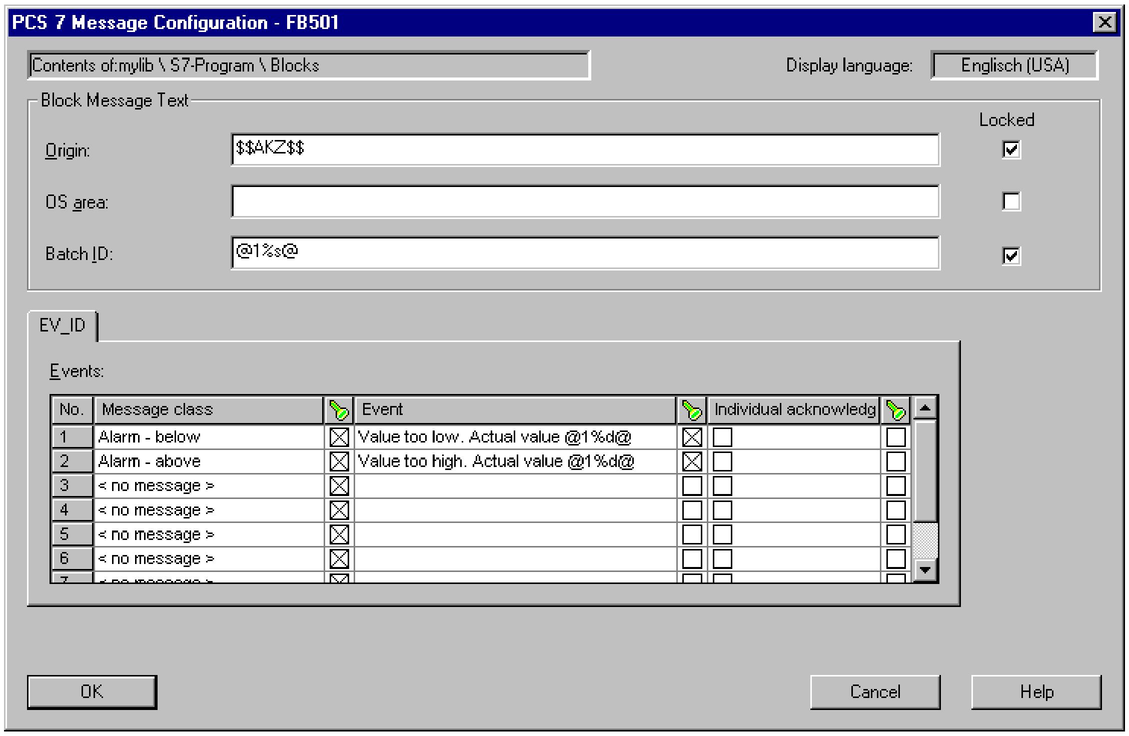
If you want your block to send messages to the OS, you must define the multiple instance of an alarm block in the static variables.

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 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.


PCS 7 Message Configuration - FB501
Contents of:mylib \ S7-Program \ Blocks
Display language: Englisch (USA)
Block Message Text
Locked
EV_ID
Events:
OK
Cancel
Help

		Locked
Origin:	$$AKZ$$	✓
OS area:		☐
Batch ID:	@1%s@	✓

No.	Message class		Event		Individual acknowledg	
1	Alarm - below	☒	Value too low. Actual value @1%d@	☒	☐	☐
2	Alarm - above	☒	Value too high. Actual value @1%d@	☒	☐	☐
3	< no message >	☒		☐	☐	☐
4	< no message >	☒		☐	☐	☐
5	< no message >	☒		☐	☐	☐
6	< no message >	☒		☐	☐	☐

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 by the PLC-OS connection configuration when the data are transferred.
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 by the PLC-OS connection configuration when the data are transferred.

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 by the PLC-OS connection configuration when the data are transferred. This is, however, not the batch ID but rather the batch name. If you want your block to be suitable for the BATCH *flexible* S7 optional package, you must enter @1%s@ here. With this entry, the message includes the BATCH batch identifier as the first auxiliary value (see also Section 1.8). If you do not use BATCH *flexible*, 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 text" column (maximum 40 characters including any auxiliary values) and whether or not the message must be acknowledged individually in the "Individual acknowledgment" column (if the check box is selected) or whether the message can be acknowledged by a group acknowledgment.

In the "Locked" column (green key symbol), you decide whether the message text can be modified by the user of the block (check box not selected) or whether the message text is locked (check box selected).

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:

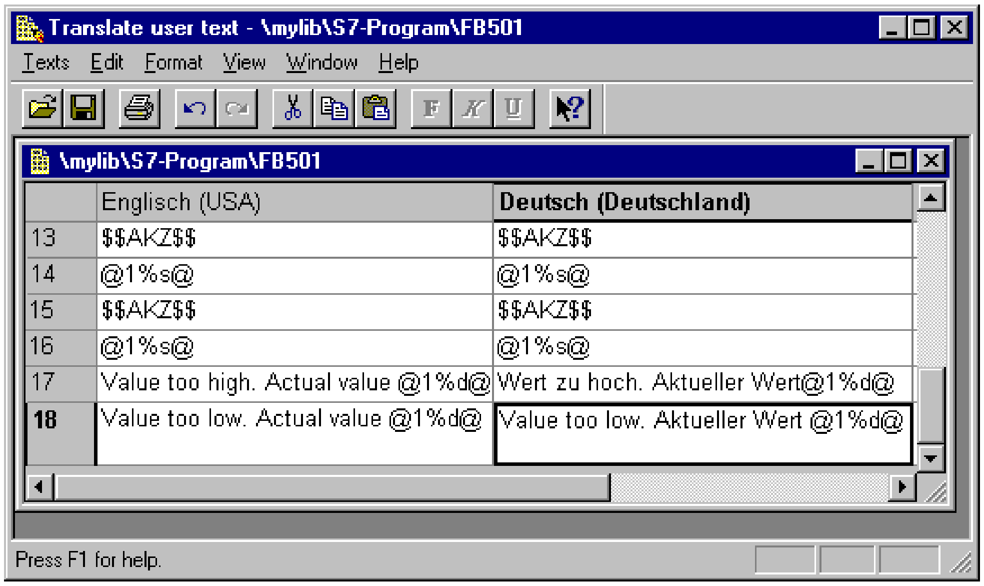
@ Parameter number format statement @

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@

Languages

If you want your messages displayed in various languages (depending on the language selection in ALARM Logging in WinCC), you must configure them for each required language by selecting the "***Options >Translating Text > User texts***" dialog in the SIMATIC Manager. Enter the texts you require in the relevant language columns in the table displayed.


Translate user text - \mylib\S7-Program\FB501
Texts Edit Format View Window Help
\mylib\S7-Program\FB501
Press F1 for help.

	Englisch (USA)	Deutsch (Deutschland)
13	$$AKZ$$	$$AKZ$$
14	@1%s@	@1%s@
15	$$AKZ$$	$$AKZ$$
16	@1%s@	@1%s@
17	Value too high. Actual value @1%d@	Wert zu hoch. Aktueller Wert@1%d@
18	Value too low. Actual value @1%d@	Value too low. Aktueller Wert @1%d@

Figure 1-8: Translating Message Texts

1.8 Linking BATCH *flexible*

If you want to use your blocks with the "BATCH *flexible"* 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 batch name	INPUT	STRING [16]
BA_ID	Consecutive batch number	INPUT	DWORD
OCCUPIED	BATCH occupied ID	INPUT	BOOL
STEP_NO	BATCH step number	INPUT	WORD

Excerpt from the sample block:

```
//****************************************************************************************
// Declaration Section: Block Parameters
//****************************************************************************************
VAR_INPUT
....
                   // Parameters for BATCH flexible
  STEP_NO {S7_visible := ’false’;
      S7_m_c   := ’true’}
            :WORD;                 // 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, but
                                   // is only monitored for binary change
     }      :BOOL := 0;            // Batch enable

  BA_NA {S7_visible := ’false’;
     S7_m_c   := ’true’}
      :STRING[16] := ’’;           // 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 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. This is therefore a typical application of the multiple instance technique.

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 (FB601).

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** (FC63) to the chart.
- Interconnect the "OUT" output of **MUL_R** with the process value (parameter "PV_IN") of the sample block.
- Open the view of the chart I/Os "Chart Inputs/Outputs" in the chart and select the symbol "IN" in the "Interface" window.
- 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).
- Interconnect all inputs and outputs of the sample block (except for the process value that is already interconnected) 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: 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 every OB contained in the task lists of the blocks it contains; in other words, its task list is made up of the task list of its blocks. The blocks contained in the block type are themselves 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 OB80 and OB100 but not **MUL_R**.

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:                   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:      '0.01'
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)

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

  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 PLC)
      }
                  :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 PLC)
      }
                 :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

  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 BATCH flexible
  STEP_NO {S7_visible := 'false';
      S7_m_c   := 'true'}  :WORD;     // 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[16] := '';  // 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
 PV_IN {S7_dynamic:='true';
    S7_m_c:='true';
    S7_unit:='%'} :   REAL := 0;     // Process value (to AUX_PR04 of message)

 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

  swSTEP_NO   :WORD;       // Batch step number
  sdBA_ID    :DWORD;       // Batch ID

  sbyBA_NA    :ARRAY[1..16] OF BYTE := 16(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
   SP_OP := OP_A_LIM_1.U;     // Write back setpoint
   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 (LMN > H_ALM) THEN   // If the manipulated variable violates the high alarm limit
```

```
     QH_ALM  := 1;     // set error output
     QL_ALM  := 0;     // reset error output

   ELSIF (LMN < L_ALM) THEN  // If the manipulated variable 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 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)

 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:= M_SUP_AH AND QH_ALM,  // Signal to be monitored 0 -> high alarm message
      SIG_2:= M_SUP_AL AND QL_ALM,  // Signal to be monitored 1 -> low alarm message
      SIG_3:= 0,           // Signal to be monitored 2 -> no message
      SIG_4:= 0,           // Signal to be monitored 3
      SIG_5:= 0,           // Signal to be monitored 4
      SIG_6:= 0,           // Signal to be monitored 5
      SIG_7:= 0,           // Signal to be monitored 6
      SIG_8:= 0,           // Signal to be monitored 7
      SD_1 := sbyBA_NA,      // Auxiliary value 1
      SD_2 := swSTEP_NO,     // Auxiliary value 2
      SD_3 := sdBA_ID,      // Auxiliary value 3
      SD_4 := PV_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

Requirements and Previous Experience

The faceplates described here are intended for use in WinCC. To create the blocks, you require the basic WinCC package with 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 Steps in Creating a Faceplate

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

- Designing the faceplate (Section 2.1.1)
- Configuring the faceplate (Section 2.1.2)
- Testing the faceplate (Section 2.1.3)

2.1.1 Designing the Faceplate

Appearance

A faceplate is the operator control and monitoring interface to a PLC block. There are two ways in which a faceplate can be displayed:

- **Group display:** Display of the PLC 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

Which input, output, and in/out parameters of a PLC block can be controlled and monitored is specified based on the system attributes when the PLC block is created. For details of the system attributes, refer to the description of the declaration section in "Structure of a PLC Block".

Parameters

The parameters are selected according to the following criteria:

- Which data do the operators require to recognize the current status both 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? You should 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 specifying the parameters and their display, you design the faceplate; in other words, select the picture elements, their names, the setting of parameters for the picture elements, and their positions. You should 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 rule makes both project documentation and maintenance easier.

2.1.2 Configuring the Faceplate

When configuring the faceplate, you can use the "WinCC Graphics Designer" tool. Starting with the templates provided by the Faceplate Designer, you convert your picture design into WinCC pictures. For a detailed description, refer to Section 2.2 "Creating Faceplates with the Faceplate Designer".

2.1.3 Testing the Faceplate

You should test the faceplate in two steps:

1. **Check the properties of the pictures you have created in the 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, if the bar display is not consistent with the numeric display) and increase the communication load of the system.
   - Are the direct connections correct?
   - Are all the event-triggered scripts present?
2. **Checks in WinCC run time:**
   - Does the faceplate open in the group display when you click on the block icon?
   - Does the switchover to individual views from the group display function correctly?
   - Does the faceplate open in the loop display when you click the "loop display" button?
   - Are the values of the PLC 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 PLC block following operator input?
   - Is the operator input log correct?

2.2 Creating Faceplates with the Faceplate Designer

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

Compatibility with PCS 7 Faceplates

The faceplates supplied with PCS 7 (.ocx standard faceplates) cannot be modified with the Faceplate Designer. It is, however, possible to create new faceplates with the Faceplate Designer that have the same names as the standard faceplates (for example MOTOR, VALVE, MEAS_MON, ...). With this method, the standard faceplates are then overwritten.

The "old" .ocx faceplates (symbolic display) must be replaced by the "new" block icons in the process picture , otherwise the correct faceplate cannot be called. If you have already installed ocx faceplates, these are not overwritten by the "new" ones.

The newly created faceplates can then be treated as standard faceplates during configuration and in run time.

If standard faceplates are accidentally overwritten in the project, you can copy the standard faceplates from the library back to the project by starting the Split Screen Wizard.

Further Information

For more detailed information about the components and functionality of the Faceplate Designer, refer the WinCC online help → WinCC Options → Faceplate Designer.

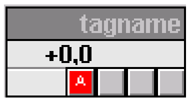
2.2.1 Templates of the Faceplate Designer

When you create faceplates, you can use the following templates that are available in WinCC:

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

You will find a list of all the files in the online help "WinCC Options > Faceplate Designer > Components of the Faceplate Designer > Faceplate Designer File List".

2.2.1.1 Block Icons



The block icons can be found in the WinCC picture "@@PCS7Typicals.PDL", for example Valve, Drive, Measured Value, Controller etc. stored in the path "Siemens\WinCC\Options\PDL\Base_Data_Pool". After running the Split Screen Wizard, this is located in the "GraCS" folder of the project folder.

- 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.

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

2.2.1.2 Template Pictures

The pictures and bitmaps are stored in the folder "WinCC\options\pdl\FaceplateDesigner".

2.2.1.3 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" and is copied to the "GraCS" folder of the project folder when you run the Split Screen Wizard.

2.2.1.4 Global Scripts

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

2.2.2 Steps in Configuration

To create a faceplate:

- Open the OS in the SIMATIC Manager.
  The WinCC Explorer opens.
- Start the Graphics Designer in the WinCC Explorer.
  In the project data, you will find the templates for creating a faceplate. The template pictures required are stored in the path

"Siemens/WinCC/options/pdl/FaceplateDesigner".

- Before you first use the pictures, run the Split Screen Wizard (in Base Data) so that the files are copied to the "GraCS" folder of the project folder.

Notes on Configuration

- If you accidentally overwrite a template file (for example, @PG_%Type%_%View%.PDL), you can copy the original from the Siemens\WinCC\Options\Pdl\FaceplateDesigner folder.
- We recommend that you initially store all files you create yourself for faceplates in the GraCS folder of the current project. If you do not want the files to be overwritten by the originals the next time you run the Split Screen Wizard, they must be copied to the Siemens\WinCC\Options\Pdl\FaceplateDesigner folder.
- If you want to use a project on a different computer, you can create a new folder with the name "FaceplateDesigner" in the project folder \GraCS. The faceplates stored in this folder are copied to the GraCS folder of the project when you start the Split Screen Wizard. Files with the same name are overwritten.
- If required, the functions configured in your own faceplates can be protected from viewing and modification 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 during configuration. The performance of a faceplate therefore depends to a large extent on the selection of suitable dynamics during configuration. In this respect, an optimized, "trim" interface between the PLC and OS functions is particularly important. This applies above all to the block icons.
- No C script with fixed coded instance names must be used in the dynamics of the faceplates.
- Different faceplate types must not be mixed in a picture used to create a faceplate; in other words, a faceplate containing, for example, a valve controller and a motor controller is not permitted.

For further information, refer to the WinCC documentation in "Configuration Notes, Tips and Tricks".

Procedure

The following sections contain a step-by-step description of how to create a finished faceplate based on the example of the MEAS_MON block type.

You create a faceplate in four consecutive steps.

3. Adapt the group display frame to the faceplate type (Section 2.2.2.1)
4. Adapt the view list to the faceplate type (Section 2.2.2.2)
5. Edit type and view-specific pictures (Section 2.2.2.3)
6. Adapt the loop display frame to the faceplate type (Section 2.2.2.4)

2.2.2.1 Adapting the Group Display Frame to the Faceplate Type

The picture "@PG_%Type%.PDL" represents the frame for the group display of a faceplate. To adapt it to the block type MEAS_MON, follow the steps below:

- Open the template picture "@PG_%Type%.PDL" in the WinCC Graphics Designer
- Specify the faceplate type by selecting the object "Block Type" (%Type%) and opening the properties dialog ("Properties" tab).
  Object properties window → Properties → I/O Field → Output/Input → Output value → "%Type%".
  Replace "%Type%" with "MEAS_MON"
- If necessary, switch the group display "EventState" invisible:
  object properties window → Properties → Miscellaneous → Display → no
  This is not necessary in the MEAS_MON example.
- Save the picture under the name of the block type "@PG_MEAS_MON.PDL"

2.2.2.2 Adapting the View List to the Block Type

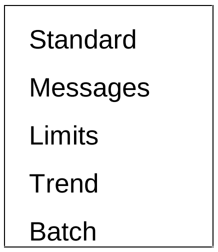
The picture "@PG_%Type%_VIEWLIST.PDL" contains all views that can be used for the PCS 7 standard faceplates V5.1.

Standard
Maintenance
Messages
Parameter
Limits
Trend
Batch
StandardS

To adapt it to the block type MEAS_MON, follow the steps below:

- Open the picture "@PG_%Type%_VIEWLIST.PDL" in the WinCC Graphics Designer
- Remove the views you do not require. For MEAS_MON, these are: "Maintenance", "Parameters" and "StandardS" (select and delete).
- Place the entries in the required order without any gaps by selecting the entries in the list and moving them with the arrow buttons.
- Adapting the frame and picture size:
  - For the picture size, open the properties dialog of the "@PG_%Type%_VIEWLIST" picture object. In Properties/Picture Object/Geometry, you can make the required static modifications (for example, picture width 90, height 100).
  - Now open the properties dialog of the "Comboframe" object. In Properties/Rectangle/Geometry, you can now adapt the frame (to fit the picture size).

The picture then appears as shown below:



Standard
Messages
Limits
Trend
Batch

- Save this picture with the block type name, in our example, as "@PG_MEAS_MON_VIEWLIST.PDL".

2.2.2.3 Editing Type and View-Specific Pictures

The picture "@PG_%Type%_%VIEW%.PDL" is the template for a view of the faceplate.

For the alarm, trend, and batch view, the pictures "@PCS7_ALARM.PDL", "@PCS7_TREND.PDL", and "@PCS7_BATCH.PDL" are available. These are identical for all faceplates in PCS 7 and they need to be adapted only to meet special requirements. Change the %VIEW% placeholder for other views. The views must adhere to the following naming convention:

Purpose	**Placeholder**	**Name**
Limits	%VIEW%	= LIMITS
Parameter	%VIEW%	= PARAMETERS
Standard	%VIEW%	= STANDARD
Standard S	%VIEW%	= STANDARDS
Maintenance	%VIEW%	= MAINTENANCE

To adapt it to the block type MEAS_MON, follow the steps below:

- Open the picture "@PG_%Type%_%View%.PDL" in the WinCC Graphics Designer
- Save this picture for the **standard view** as "@PG_MEAS_MON_**STANDARD**.PDL".
- Open the picture "@PG_%Type%_%View%.PDL" in the WinCC Graphics Designer
- Save this picture for the **limits view** as "@PG_MEAS_MON_**LIMITS**.PDL".

Changing the picture size

If you change the size in the Properties dialog (Properties > Geometry), remember the following points:

- The views of the pictures are displayed in the group display and in the loop display.
- At the present time, universally valid pictures are used for Alarm, Batch, and Trend. Names, see above.
- If you change the size of the other views, you may also need to modify these universal pictures.

2.2.2.4 Adapting the Loop Display Frame to the Block Type

The picture "@PL_%Type%.PDL" represents the frame for the loop display of a faceplate. To adapt it to the block type MEAS_MON, follow the steps below:

- Open the picture "@PL_%Type%.PDL" in the WinCC Graphics Designer
- In the I/O field of the object type "BlockType" replace the text "%Type%" with the block type name:
  Object properties window → Properties → I/O Field → Output/Input → Output Value → "%Type%"
  Replace "%Type%" with "MEAS_MON"
- If necessary, switch the group display "EventState" invisible:
  object properties window → Properties → Miscellaneous → Display → no
  This is not necessary in the MEAS_MON example.
- Enter the required view in the "@VIEWS" object

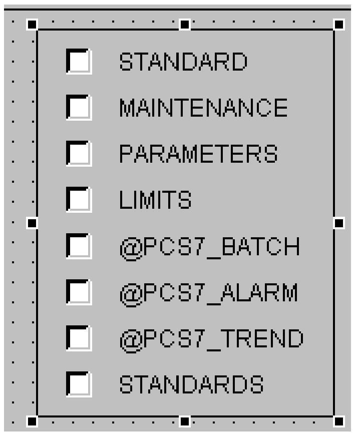


Figure 2-1 : "@VIEWS" Check Box

- Object properties window → Properties → Check Box → Output/Input. Double-click on "Selected Boxes".
- Click on the first required view (for example, "1" for STANDARD).
- Click the index of the view for the other required views while holding down the CTRL key.
- Confirm your selection with the OK button.
- Check whether the required views are shown as "selected" (check mark in the appropriate check box).
- Remove the views you do not require (for MEAS_MON, these are the elements "MAINTENANCE", "PARAMETERS" and "STANDARDS".
- Place the elements in the required order in the picture and adapt the picture size to the size of the picture window.
- Save the picture under the name of the block type "@PL_MEAS_MON.PDL"

2.2.2.5 Configuring the Trend View

In the steps up to now, you have only created type-specific pictures. The trend view requires additional configuration for the specific tags. The trend view "@PCS7_TREND.PDL" available for all PCS 7 faceplates contains the picture window "TrendPicture". To include a trend picture for a tag at this point, follow the steps below:

- Open "Tag Logging" in the WinCC editor.
- Create a tag archive for the trend view using the Archive Wizard
- Create the tags for this archive

For detailed instructions explaining the steps above, refer to "Tag Logging" in the WinCC online help.

- Open the picture "@CONL1_.PDL" in the Graphics Designer
- Configure the TlgOnlineTrend object "Control1" according to your requirements; you can get more information with the "Help" button.
- Save the picture as "@CONL1_<tagname>.PDL". The <tagname> must be identical to the tag name of the PLC block. You must replace the character "/" in the name with "_", since Windows does not accept "/" in file names.

  Example of the window name of the MEAS_MON_1 block:
  @CONL1_Measurements_Opinputs_MEAS_MON_1.PDL

2.2.2.6 Configuring Further Views

In configuration step 3 (Section 2.2.2.3), you already created the views for a faceplate type. The picture elements are now inserted in the prepared views and assigned parameters according the picture design. The finished samples "@PG_MEAS_MON_STANDARD.PDL" and "@PG_MEAS_MON_LIMITS.PDL" are on the CD so that at this point, only a general outline of the steps is necessary. You can find out which objects are installed in the pictures and see the properties of the objects in "Object Properties".

For the "Limits" view of MEAS_MON, follow the steps below:

- Open the picture "@PG_MEAS_MON_LIMITS.PDL" in the WinCC Graphics Designer.
- Open the picture "@PCS7Elements.PDL" in the WinCC Graphics Designer.
- Arrange the pictures one beside the other: Window > Tile Vertically.
- Copy the required picture elements from "@PCS7Elements.PDL" to "@PG_MEAS_MON_LIMITS.PDL".
- Include other WinCC objects such as rectangles in your picture.
- Assign meaningful (object-related) object names.
- Set the positions of the individual objects.
- Interconnect the dynamic attributes of the picture elements with the parameters of the PLC block type. The interconnection has already been configured for the objects "PCS7_MSG_LOCK" and "PCS7_OCCUPIED".
- Set up the "authorization sequences". The template pictures include the objects "@Level5" and "@Level6". Both objects are influenced by the control of the higher-level picture "@PG_MEAS_MON.PDL" or "@PL_MEAS_MON.PDL".

  If a user is logged in with authorization level 5, the background color "white" is set for the "@Level5" object and the operator enable is set to "yes". If the user does not have these rights, the background color is "gray" and the operator enable is "no".
  The "@Level6" object is used to control the operator authorization level 6. Both elements serve as "anchors" for all the operator-controlled objects in your picture. After the events "background color" and "operator authorization", direct connections are established from one element to the next. No direct connection is necessary for binary variables.

  In the "@PG_MEAS_MON_LIMITS.PDL" picture, the sequences for operator authorization level 6 are as follows:

  "**Operator authorization**": @Level6 > U_AH > U_WH > U_WL > U_AL > MO_PVHR > MO_PVLR > M_SUP_AH > M_SUP_WH > M_SUP_WL > M_SUP_AL

  "**Background color**": @Level6 > U_AH > U_WH > U_WL > U_AL > MO_PVHR > MO_PVLR

- Save the picture.

The steps for the "Standard" view of MEAS_MON are analogous.

2.2.2.7 Creating Your Own View of a Block Type

The description below explains how you can create a new view that applies to only one block type from a view that was previously used for all PCS 7 blocks. In this example, we are using an alarm view.

- Save the original picture "@PCS7_Alarm" as "@PG_%Type%_Alarm" (type = MEAS_MON).
- Copy the objects "@Level5" and "@Level6" to your picture from an existing view, for example, from "@PG_ MEAS_MON_Standard.PDL".

  If you lose the "@" characters when you copy, they must be inserted again so that they precede the names of the two objects Level5 and Level6.
- The following script must be inserted for "@Level5" below the C action:

  ```
  double dToolbarButtons;
  if (value)
   dToolbarButtons = 62000;
  else
   dToolbarButtons = 61952;
   SetPropDouble(lpszPictureName, "AlarmList",
         "ToolbarButtons", dToolbarButtons);
  ```

  Tip: Simply copy this text from the documentation.
- Open the "@PG_%Type%_VIEWLIST.PDL" picture and increase its size so that there is room for a new view to be added as text at the bottom.
- Copy a text box, for example "Standard", insert it and give the view a new name.

  The object name must match the new view name, in this case, "Alarm".
- Modify the display text for the view list; this can differ from the picture name and can be entered in more than one language (see Graphics-Designer > View > Language menu).
- Open the loop display "@PL_%Type%.PDL" and add an additional field to the existing check box:
  In the "@Views" properties of the check box, select "Geometry" and increase the number of fields by 1 (for example, from 8 to 9).
- This new field must be named after the new view:
  Properties > Text > Index: Enter the number of the field you have added (here: 9) and enter the name (here: "Alarm") in the "Text" field .
- To make the field visible later in run time:
  Select "@Views" > Output/Input in the properties and double-click on "Selected Fields". Select the index "9" in the list and close with "OK". Note: You can make multiple selections by holding down the CTRL key.
- Insert a picture window for the new view in the loop display frame:
  Copy the "picture window" object from an existing view (for example, STANDARD) and give it a new name (here: "Alarm").

2.2.3 Preparing Faceplates for Dynamic Display

There are different ways of achieving a dynamic display:

1. The extensions of the required variables are known.

   Open the object properties of the object you want to display dynamically (for example, a bar). Double-click the bulb symbol for the required attribute in the "Dynamics" column of the Properties window. You can now enter the extension in the input field.

2. Tag from the tag list

   The normal method, for example with process pictures, is to use the tag list. However, this lists all tags. Right-click on the bulb symbol in the Properties window and select "Tag".

   Locate the required tag and double-click on it.
   In the dynamic display, however, you now have the full tag name. Delete the text up to the extension, including the dot (".").

Procedure for Dynamic Display

- Open the view you want to display dynamically, for example, "@PG_MEAS_MON_STANDARD.PDL".
- Switch the message status display invisible if you do not require it by selecting the "EventState" object and opening the properties dialog.

  Change the "Display" attribute to "No" using properties/group display/miscellaneous.
- Install the objects in the picture and assign the parameters to them: For standard faceplates, the necessary objects are already prepared and can be copied directly from the template files to the picture, for example, copy bars from the "@PCS7Elements.PDL" and insert them in the picture. You must then interconnect the properties of the objects. If the target names are known, you can enter these directly. Otherwise, you can establish the link using the tag browser. Remember that the name of the PLC instance must not be included in the tag name but only the extension. For example, in the tag name "Plant1_Motor2.U_AH" the name of the instance "Plant1_Motor2." must be deleted (including "."). The name is then "U_AH". When using the extension, remember that it is case-sensitive. To improve performance, the update cycles should be identical whenever possible (standard = 2 sec.).

- Starting with the objects "@Level5" and "@Level6", set up a chain of direct connections for the individual authorization levels.

  Example: If you want to protect several I/O fields with authorization level 6, the "Operator Control Enable" property of the object "@Level6" must be connected directly with the "Operator Control Enable" property of the first I/O field.

  - Select the first I/O field and open the Object Properties dialog. With the "Event" tab, you can now configure the direct connection to the object "@Level6".
  - The second I/O field must now be connected directly with the first, etc.

  During configuration, remember that added operator interface elements are always added at the end of the authorization sequence.
  It is advisable to document the order of the authorization sequence.

  The change in the background color for operator control rights can be chained in the same way.

- Save the picture with the appropriate type name, here: "@PG_MEAS_MON_STANDARD.PDL".

2.2.4 Changing Languages

The templates of the Faceplate Designer are in three languages (German/English/French). This means that after you change to another language in WinCC, the texts will be displayed in the selected language. If you add text elements to a picture, and want to be able to change to one of the other languages, then remember to enter these texts in the Graphics Designer in all the languages you require. You can change to another language in the Graphics Designer with the **View > Language... > Select Language** menu command.

3 Creating Online Help

Requirements

You require the following:

- The "Notepad" ASCII editor integrated in WINDOWS NT 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 MAP ID 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)
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 NT 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 3 language versions

```
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"

[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.

Then enter the following key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\STEP7\2.0\Applications\s7libs\NameOfLibrary \Author\Family]

Below this, you must specify the MAP ID for the help file, for example "CONTROL"=dword:00000888. Where **Family** stands for the name you specified for the FAMILY attribute in the block header (here: XYZ). The MAP ID is the number of the topic ID.

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 NT registry.

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 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 symbol table of the library.
3. Source files:
   If you want to ship the source files as well, copy the source files from the sources folder of your project to the sources folder of the library.
4. Copy your blocks from the block folder of your project to the block folder of the library.
5. 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.

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 any subfolder in the **Options\Pdl** 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.
- 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 "5.1"
and in the key **HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\WinCC\Setup**
the name **Version** must be set to the value "R 5.1".

Glossary

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.

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.

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.

Declaration section

Part of a block defining the interface of the block and the data it uses internally.

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.

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.

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.

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.

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.

Library

Software package with ➔ *PLC blocks* and/or ➔ *faceplates* grouped according to common features.

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.

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.

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.

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.

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.

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.

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.

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.

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.).

WinCC

Windows Control Center: A software package for plant-oriented graphic development of ➔ *faceplates* and for operator control and monitoring of the PLC.

Index